NOKIA

Nokia 9000/9110 Development Environment (N9kDE)

Tutorial

Last Modified: June 10, 1999

Table of Contents

1 About this Document

This document covers three tutorials designed to help you use the Nokia 9000/9110 Development Environment (referred in this document as N9kDE or SDK). The tutorials are located in the "tutorials" folder of your installation.

Tutorial 1	Introduces the procedure for creating, editing, and running a N9kDE project. Uses features in the SDK designed to speed application development.	First.prj
Tutorial 2	Creates a new project with buttons, forms, and graphic forms. Examines more SDK build features, debugging, and editor features.	Forms.prj
Tutorial 3	Create a new project that includes the Flashing Note tool. Demonstrates the Toolbox manager and more advanced application development.	Flash.prj

We recommend that you proceed through the tutorials in order (tutorial 1,2,3).

2 Product Overview

The N9kDE improves application development for the Nokia 9000 Communicator class of mobile phones. It includes the following features:

- Broad-featured interactive development environment
- Simplified interaction with the Borland C++ compiler and the Geoworks SDK
- Intuitive, highly-integrated create-edit-build-test development cycle
- Convenient access to the N9000 emulator
- Automated application shell development with forms and procedures
- Simplified application packaging with graphical packager tool
- Automated code generation
- Rich text edit control
- Capability to extend the N9KS with the Toolbox API
- Eliminates redundant programming tasks.
- Provides rapid access to GEOS documentation and help.

The N9kDE is designed to operate as an extension of the existing GEOS SDK. This extension is provided through the use of executable files developed by Geoworks. The N9kDE makes use of GEOS SDK.

The N9kDE is designed to permit technological evolution and service extensions. The SDK is designed to allow for quick deployment of complete applications. This is achieved by automating the complexities associated with packaging an application for delivery to the Nokia 9000 Communicator through a specialized packaging utility.

The SDK is designed to allow for quick compilation and debugging of applications. This is achieved by an intuitive integration into the N9KS IDE of the GEOS compiler, emulator and debugger.

In summary, the SDK is an extensible development environment providing simplified extensions to the GEOS SDK. The SDK provides compatible support to existing applications and moreover, is designed to gracefully evolve through the adoption and provision of future enhancements to the GEOS language.

3 N9kDE Project Quick Guide

Here is a quick guide to projects in the N9kDE:

1. Launch the N9kDE from your Windows Start menu
2. Go to the File->New menu and enter a project name
3. Go to the Build… Menu and choose Execute with Swat <F5>
4. Press <CTRL><F12> in the emulator window to go to the Extras applications
5. Scroll to the end of the application list to find your project
6. Press <F1> to select your project. The <F4> key means “Close”
7. Dismiss the emulator to return to the SDK

4 Common N9kDE Menu and Toolbar Features

Shortcut	Description
Within the SDK	
	Creates a new project
 (Double click)	Launchs the project .GOC file in the editor
<CTRL>F	Find
<CTRL>S	Save file
<F1>	Help for the N9kDE
	Searches GEOS Technical and programmers documentation
<F5>	Build
File… Exit	Quits the SDK
Within the Nokia 9000 Emulator	
<CTRL><F12>	Go to Extras
<F1>, <F2>, <F3>, <F4>	Menu keys

5 Tutorial 1 – First Application

5.1 Objective

The objective of Tutorial 1 is to introduce projects in the SDK. This tutorial is a step-by-step illustration to creating a project named “First” and executing it within the Nokia 9000 Emulator. Tutorial 1 takes advantage of several of the automated features in the N9kDE which speed development.

5.2 Overview

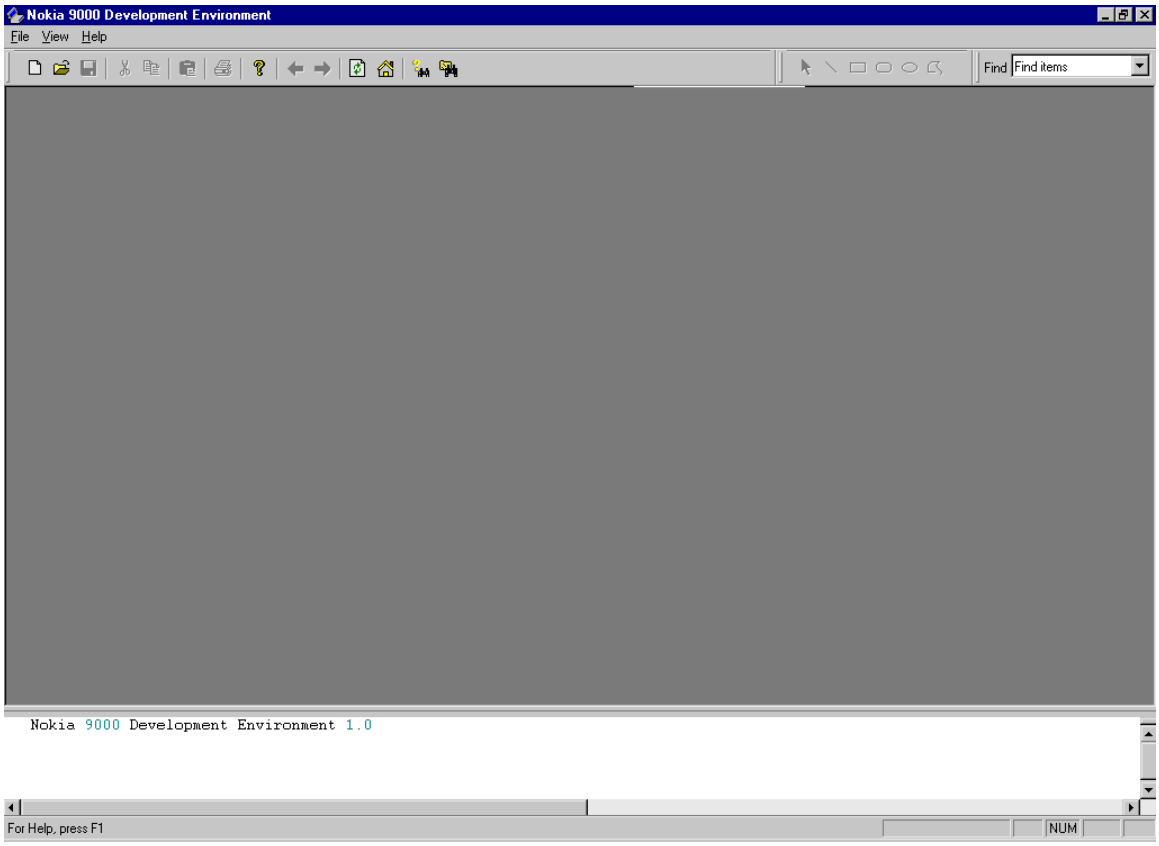
- Launch the N9kDE from your Windows Start menu
- Go to the File->New menu and enter a project name
- Go to the Build… Menu and choose Execute with Swat <F5>
- Press <CTRL><F12> in the emulator window to go to the Extras applications
- Scroll to the end of the application list to find your project
- Press <F1> to select your project in the emulator.
- Press <F4> close your project in the emulator
- Dismiss the emulator to return to the SDK
- Go to File->Exit to quit the N9kDE

5.3 Tutorial 1 Procedure

Step 1

Begin by first launching the Nokia 9000/9110 Development Environment. From your Start menu, go Programs->Nokia 9000/9110 Development Environment to launch the product.

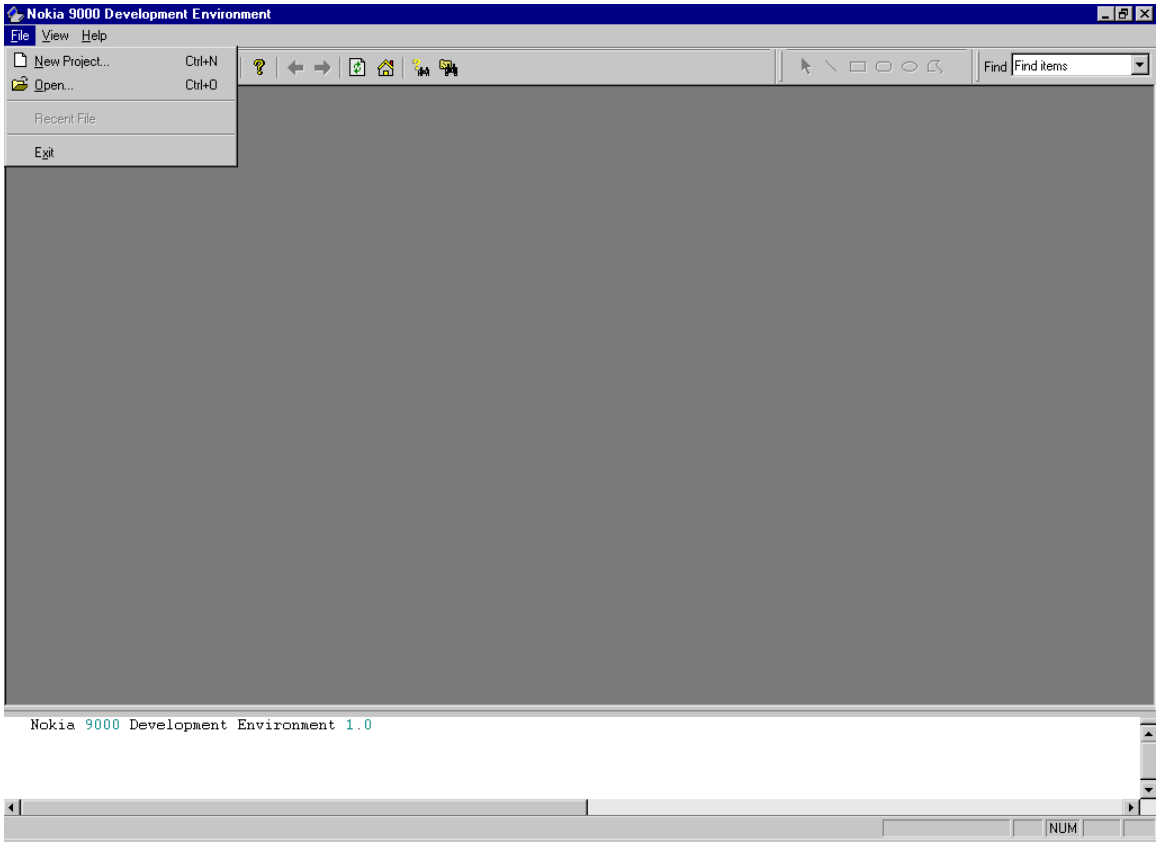
The N9kDE starts with a splash screen, followed by the workspace. The workspace is maximized to cover your full screen. Here is the Nokia 9000/9110 Development Environment workspace.



The workspace includes a menu bar, a tool bar, an output window for messages and errors, and a status bar. Many functions of the N9kDE have toolbar shortcuts and accelerator keys to speed your development. For instance, you can click on the “New” icon on the toolbar, or press <CTRL>N to create a new project.

Step 2

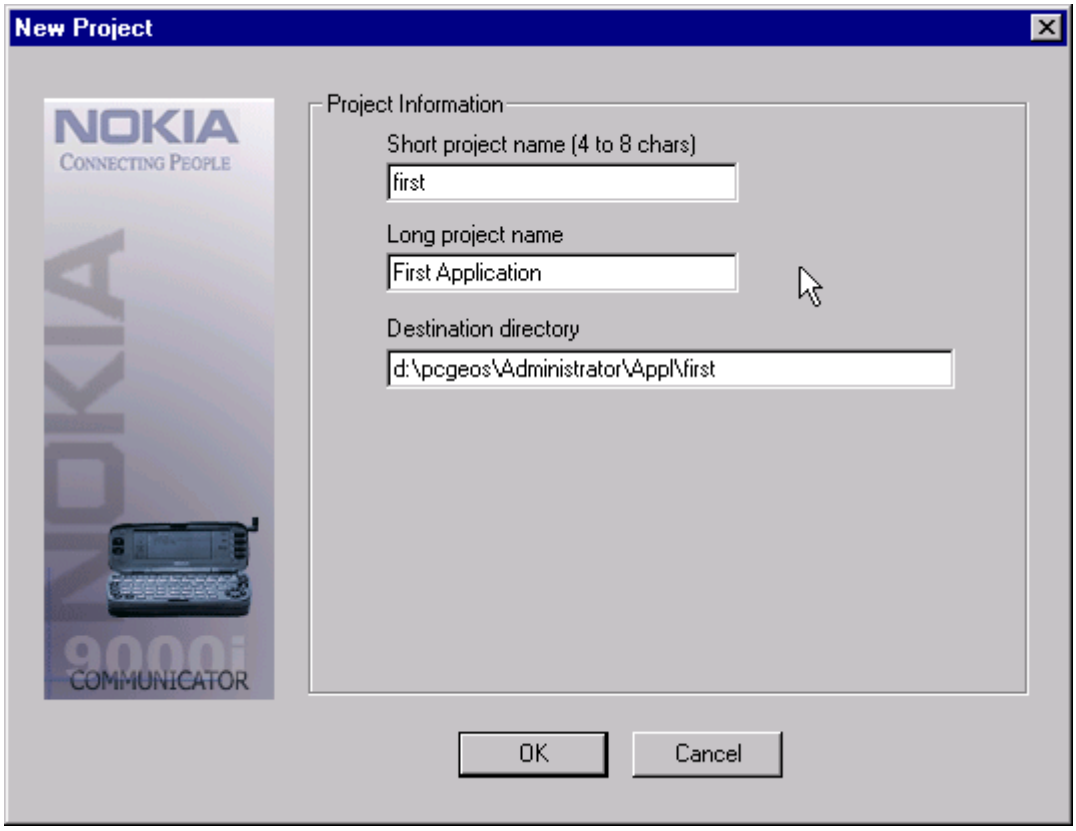
Next, create a new project by choosing File->New Project…



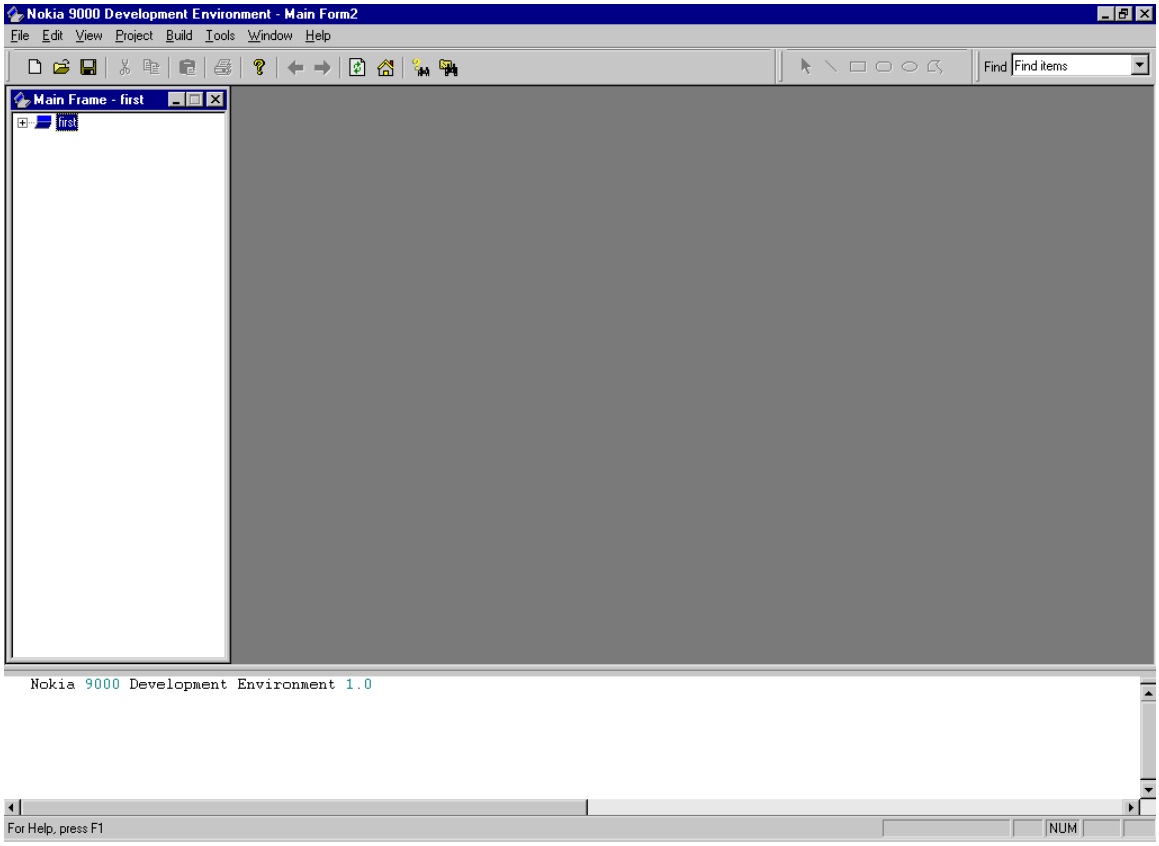
Step 3

Type “first” as the short project name and type “First Application” as the long project name. We recommend using the default destination directory. Click the “OK” button to create the project.

The short project name must have 4-8 characters and start with a letter. The short project name is used throughout the project as an identifier. The long project name is used as the title of your application. You will see the application title when you execute your project in the emulator, or when you download your application to the Nokia 9000 handset.

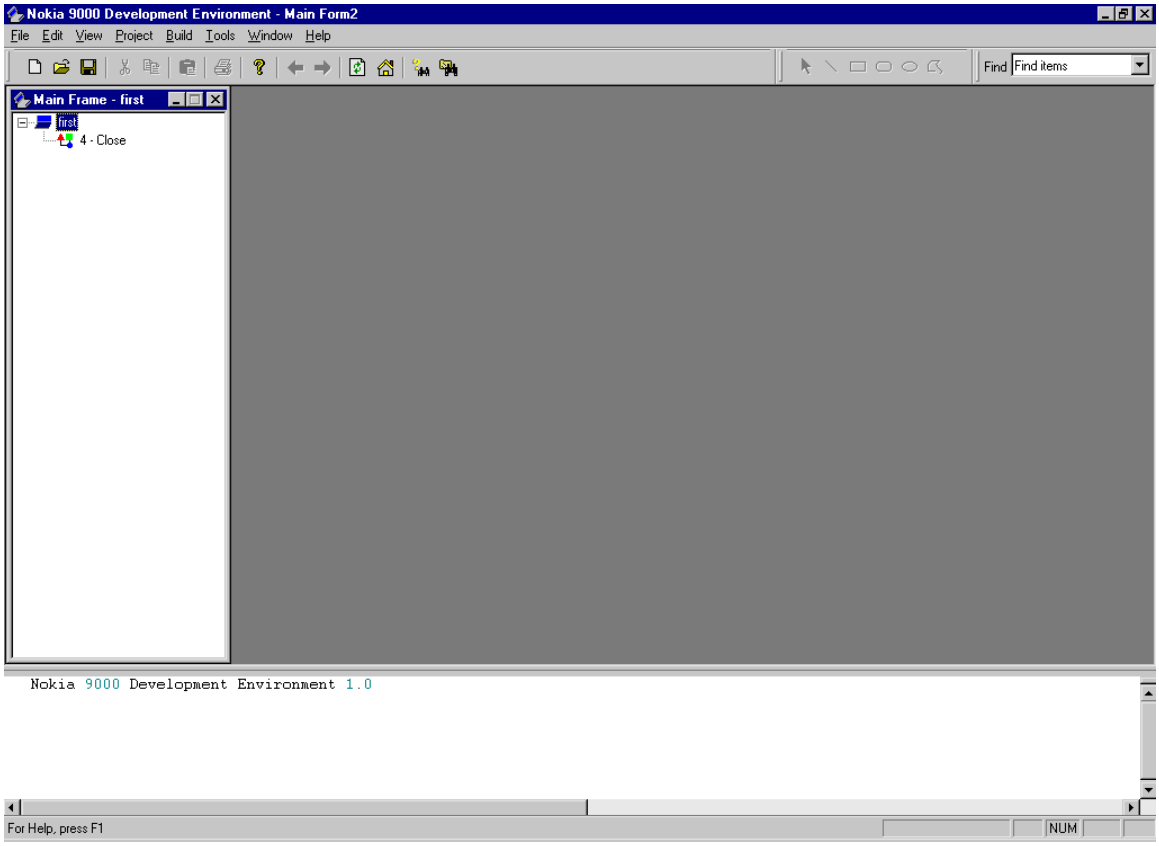


Step 5
Examine the project tree view for the your “first” application.



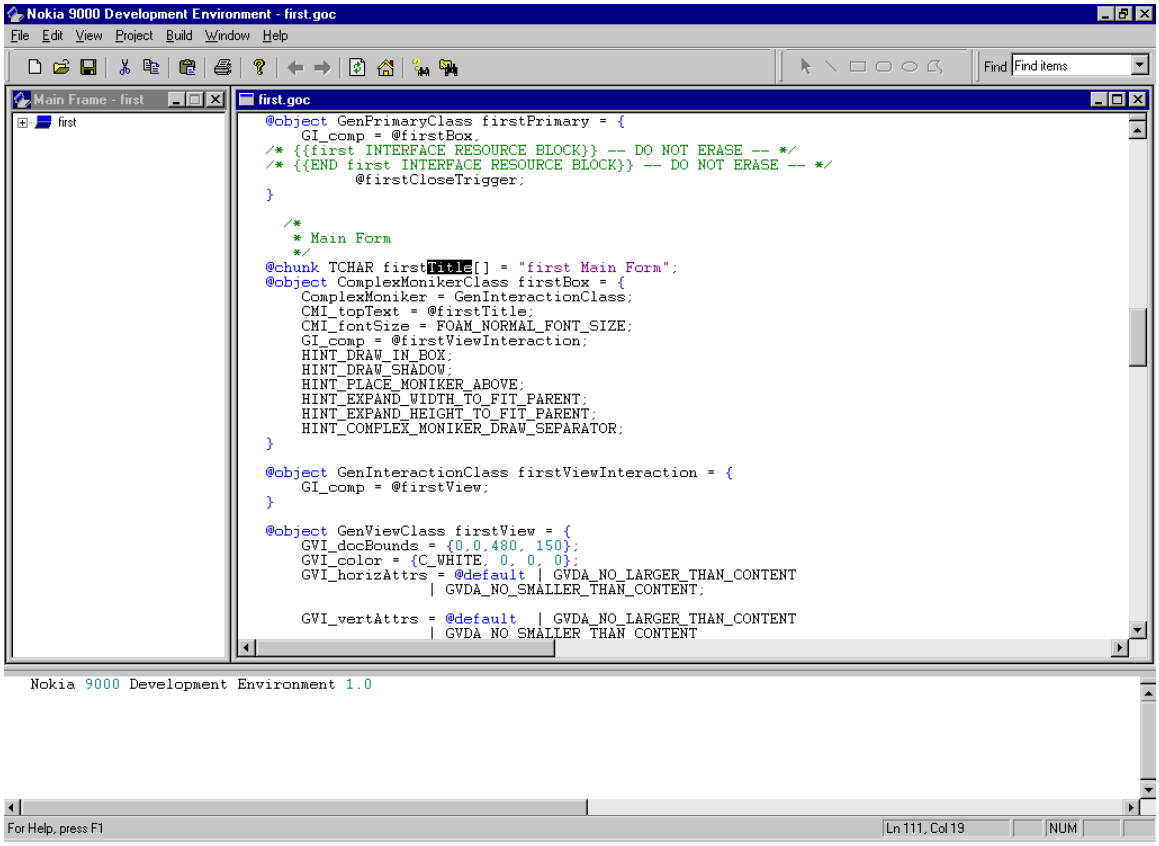
Step 6

Expand the tree view by clicking on the “+” symbol. A new project in the N9kDE includes a main form and a close button.



Step 7

Open the project GOC file to examine the code that was generated. Simply double-click on the main form icon  in the tree view. Alternatively, you can open the GOC file for the project from the File->Open menu.



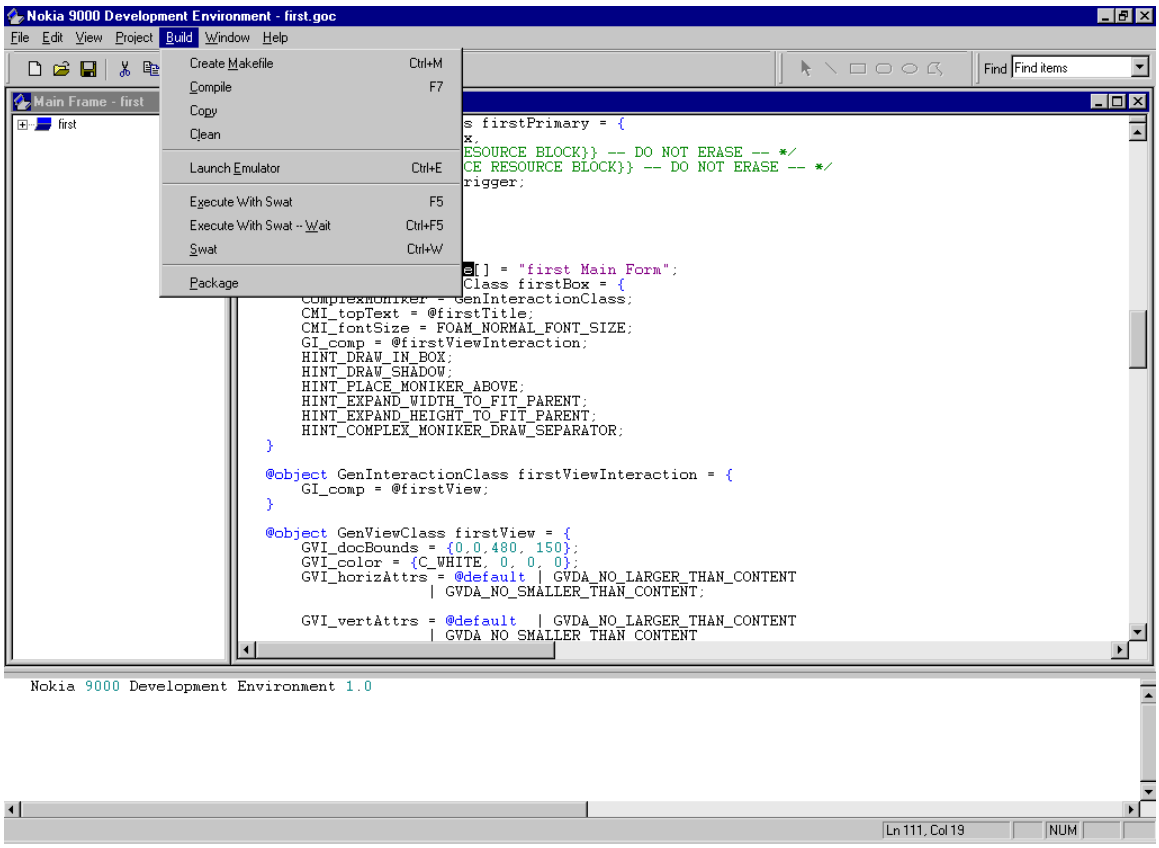
You can navigate through the source code using standard editor functions, your mouse, and arrow keys.

Step 8

Build and execute your project in the emulator by choosing Build->Execute with Swat. You can also press <F5> in the N9kDE to build (or re-build) your application and launch it in the emulator.

This option performs the complete build and launch process, which includes creating a makefile, compiling the project files, copying the .GEO file to the emulator directory, and launching the emulator.

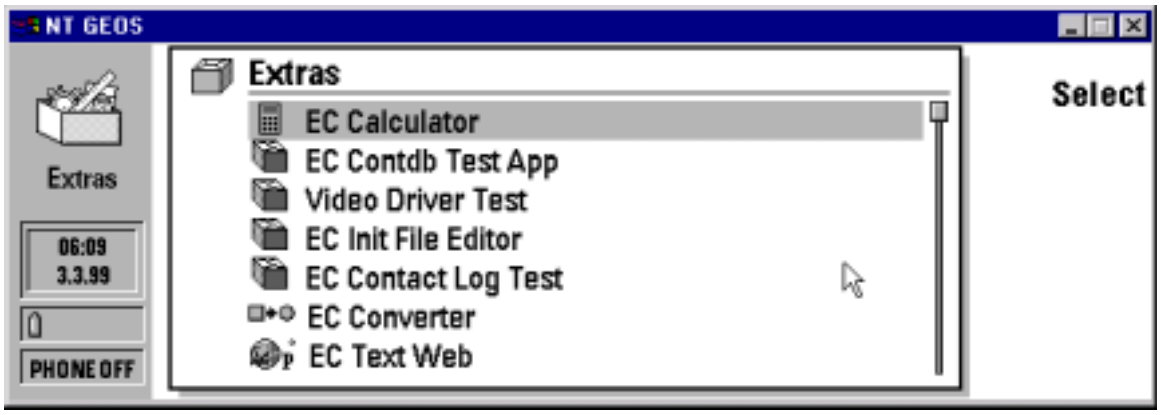
While the application builds, watch the Output window for compilation reports and other important messages. During the build process, the N9kDE runs several programs (including the Borland C++ compiler and GEOS batch files).



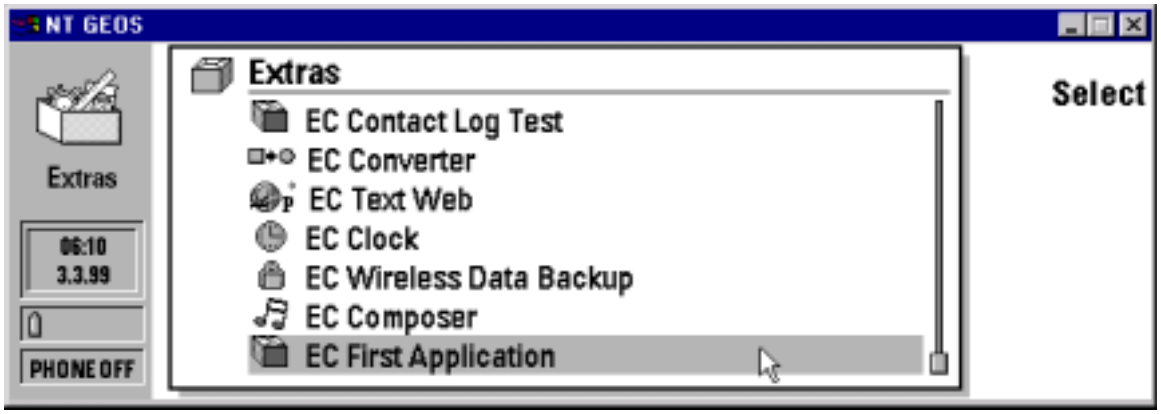
Step 9

Next, you will need to locate your application in the emulator. The emulator begins in the default program group – Telephone directory.

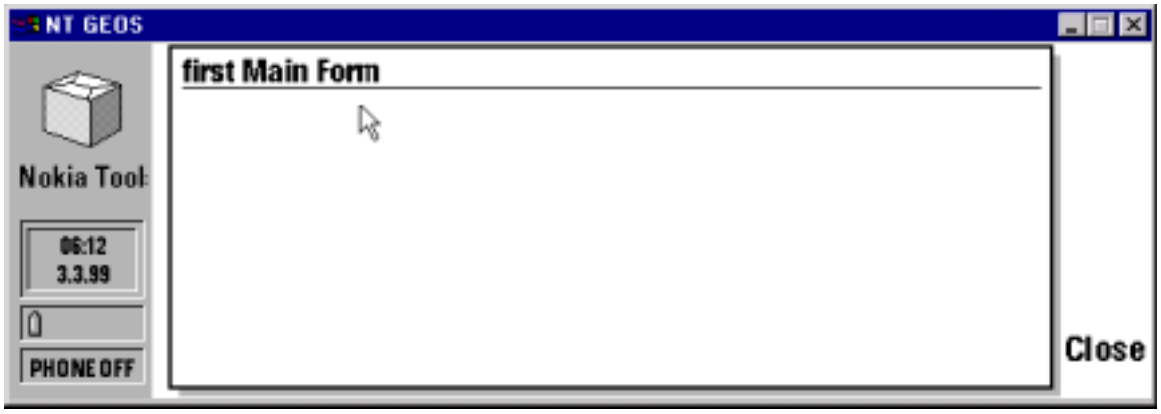
Press <CTRL><F12> to switch to the Extras, which is were your “first” application has been saved.



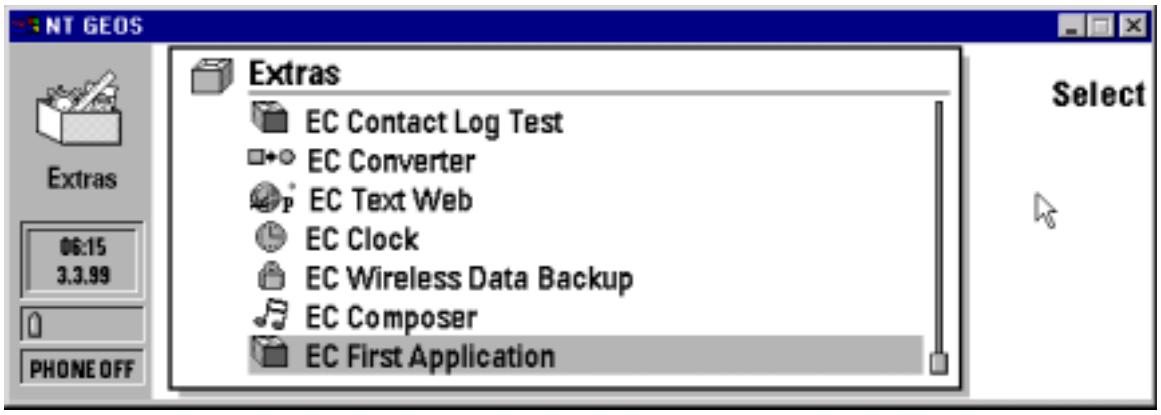
Locate your program by scrolling down the list with the arrow keys on your keyboard. The prefix “EC” has been added to your project name to indicate that you are running in the Error Correcting mode, which is the default setting for the N9kDE.



Step 10
Launch your application by pressing the <F1> key, which emulates the Select button on the handset.



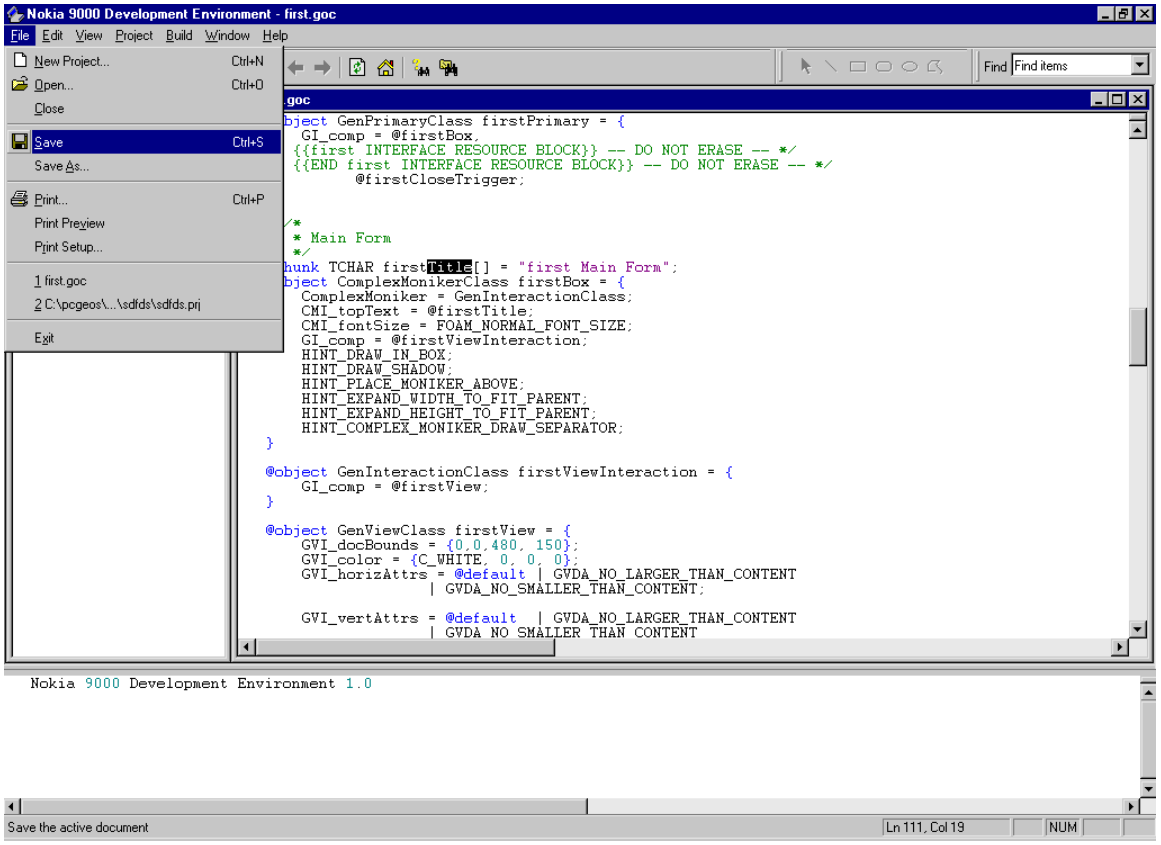
Congratulations! You have just launched your first application. Right now the “first” application is just the main form and the close button, just like the tree view displays within the N9kDE.
Step 11
Close your application by pressing <F4>, which emulates the Close button on the handset. You are returned to the Extras menu in the emulator.



Dismiss the emulator by clicking  on the emulator window.

Step 12

Next, save your project by choosing File->Save from the menubar.



You can exit the N9kDE by choosing File->Exit from the menubar.

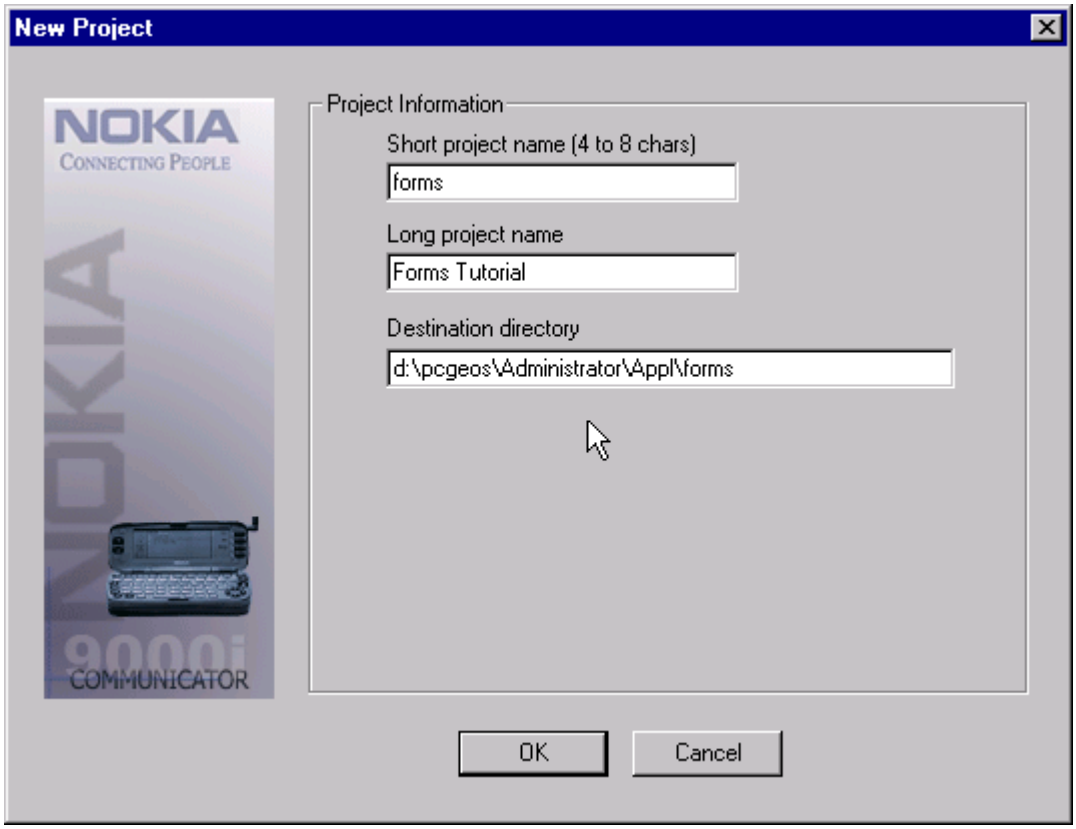
6 Tutorial 2 – Forms Application

6.1 Objective

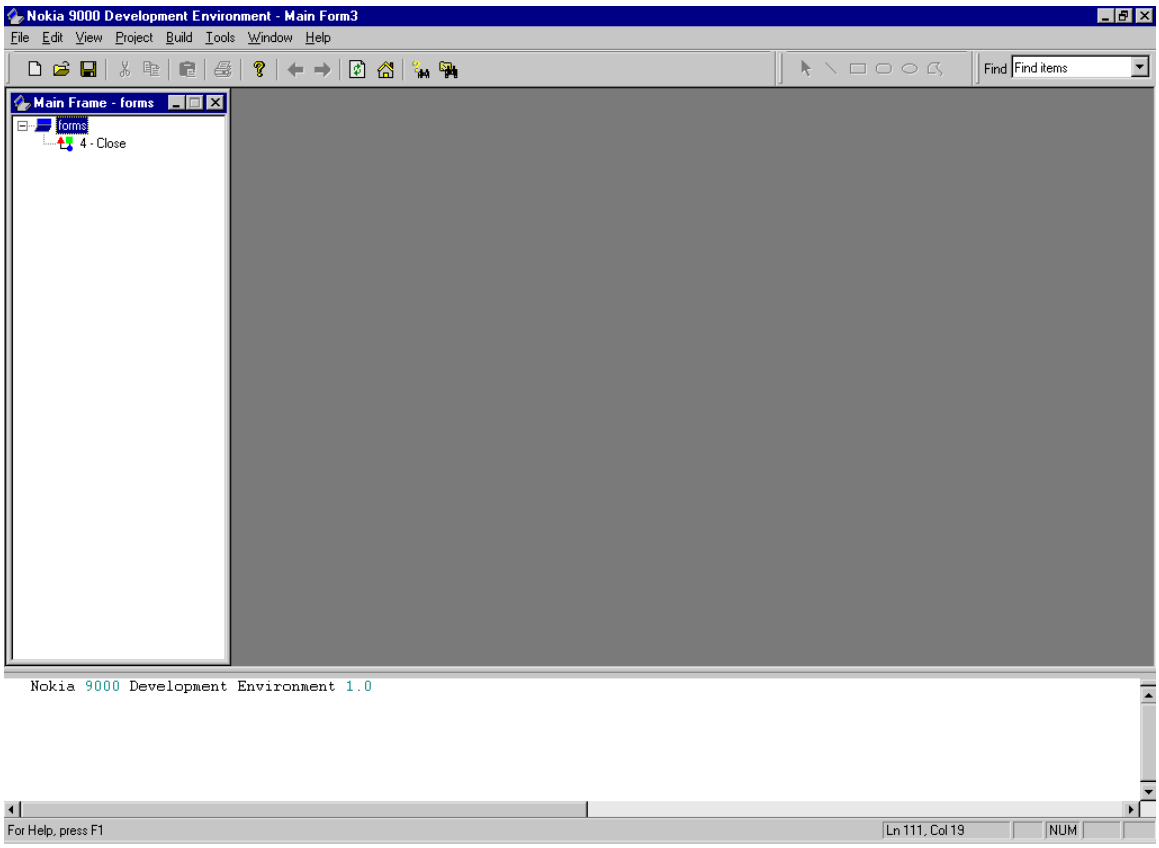
The objective of Tutorial 2 is to illustrate the form and procedure features of the SDK. The SDK makes it possible to quickly add new forms (both standard and graphic) and procedures to an application which are important rapid prototyping features. In addition, Tutorial 2 examines more SDK build features, debugging, and editor features.

6.2 Procedure

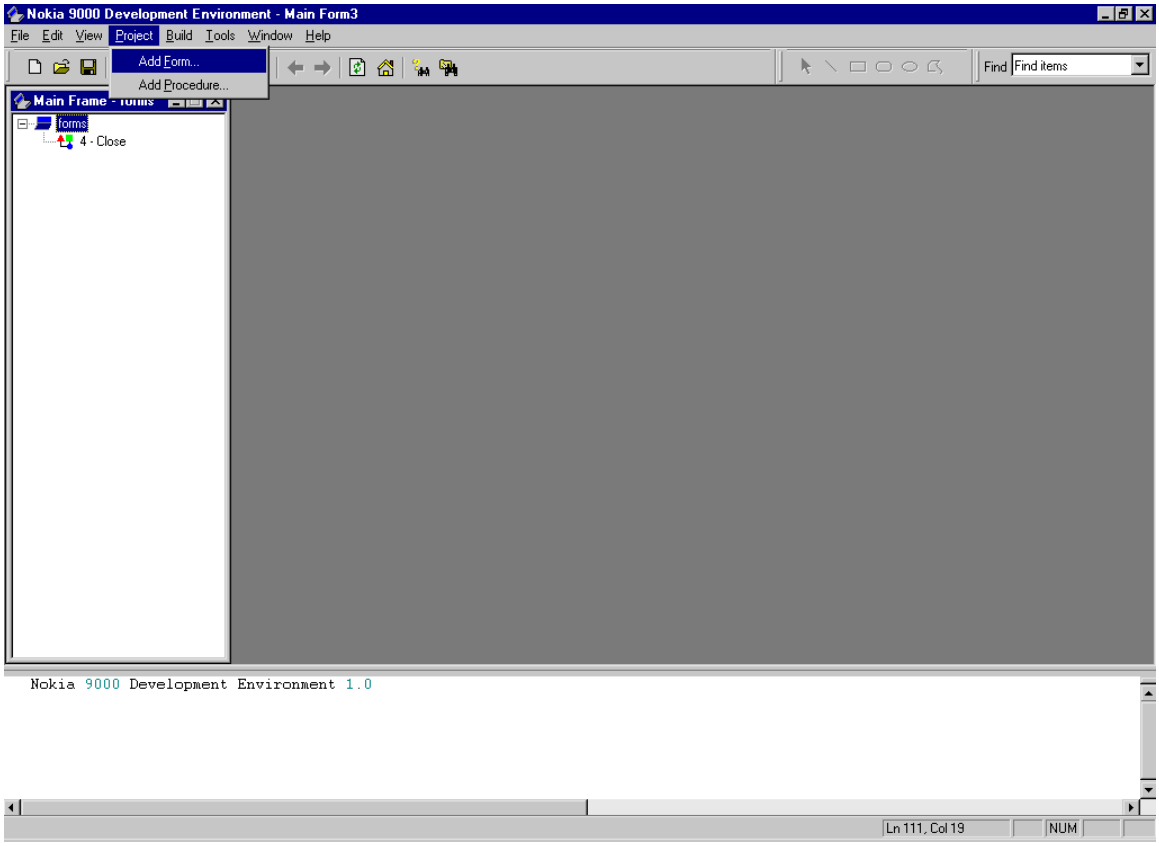
Step 1
First, create a new project in the N9kDE. The project in Tutorial 2 is named “Forms”:



Step 2
Expand the project tree view to see the main form and its close button.

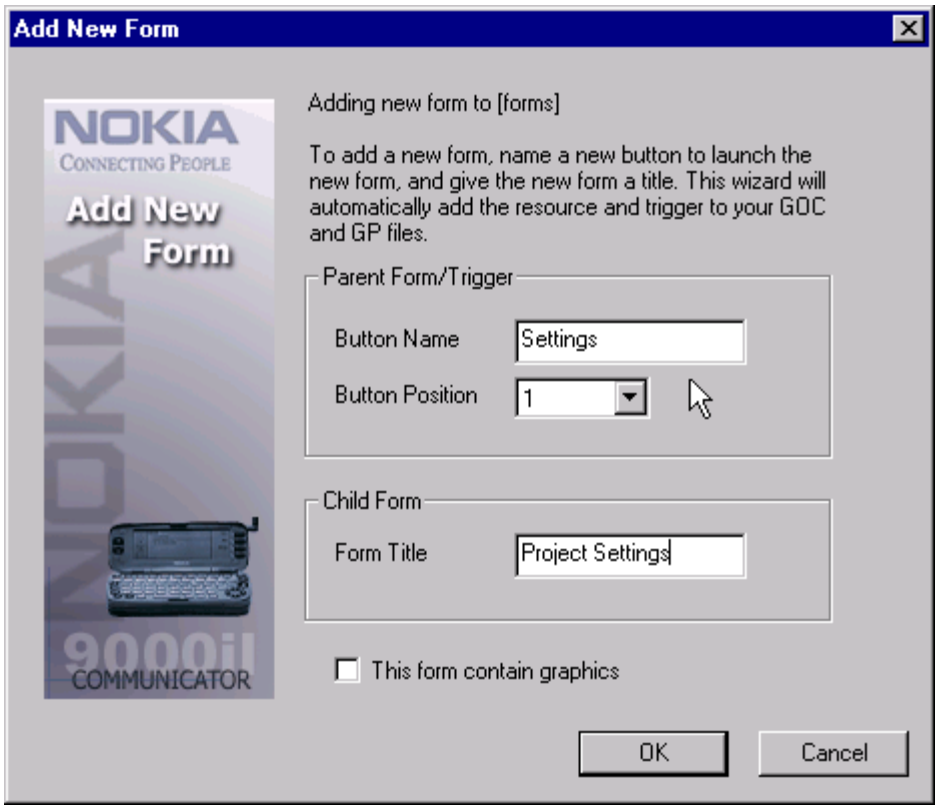


Step 3
Add a form to the project by choosing Project-> Add Form from the menubar.



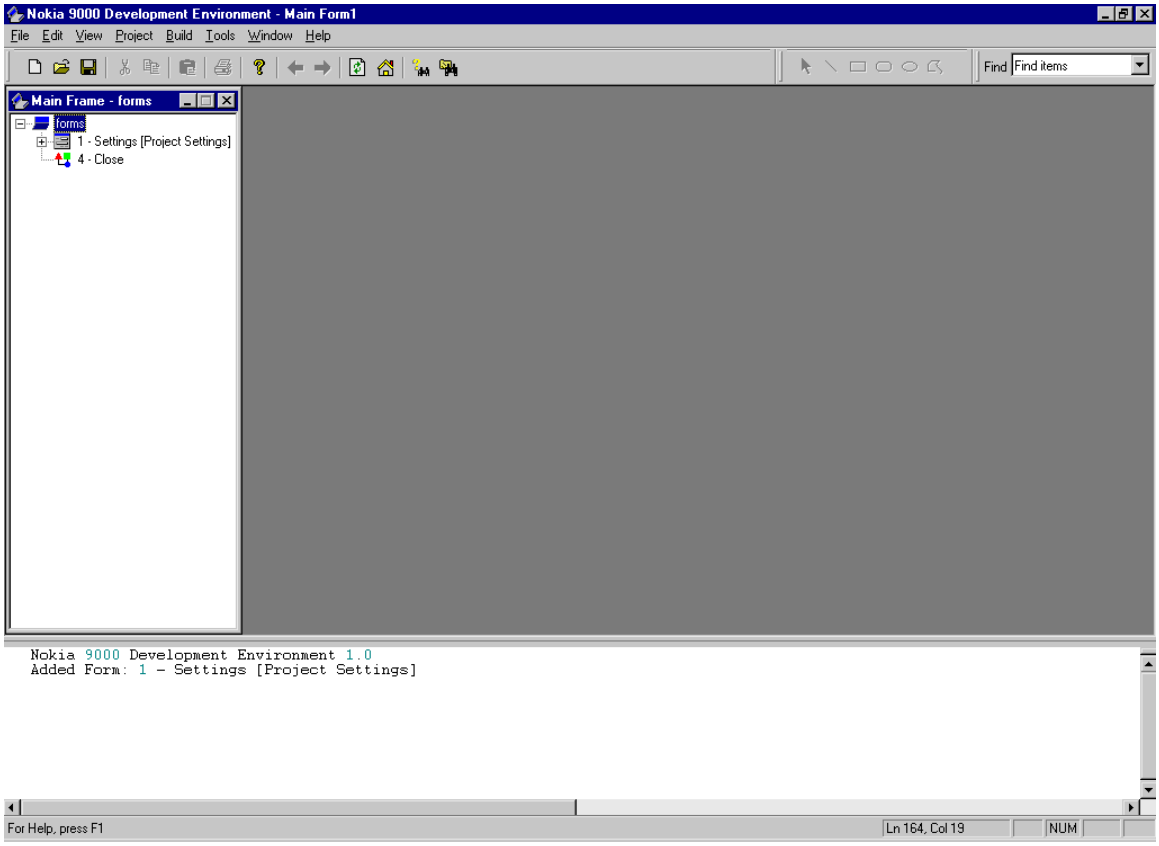
You can add both new forms and procedures to your project in the N9kDE. Each time you add a new form, you will create a button (trigger) which launches the form in your program.

- Enter “Settings” for the button name.
- Choose button position 1.
- Name the new form (child form) “Project Settings”.
- Leave the graphics “off”



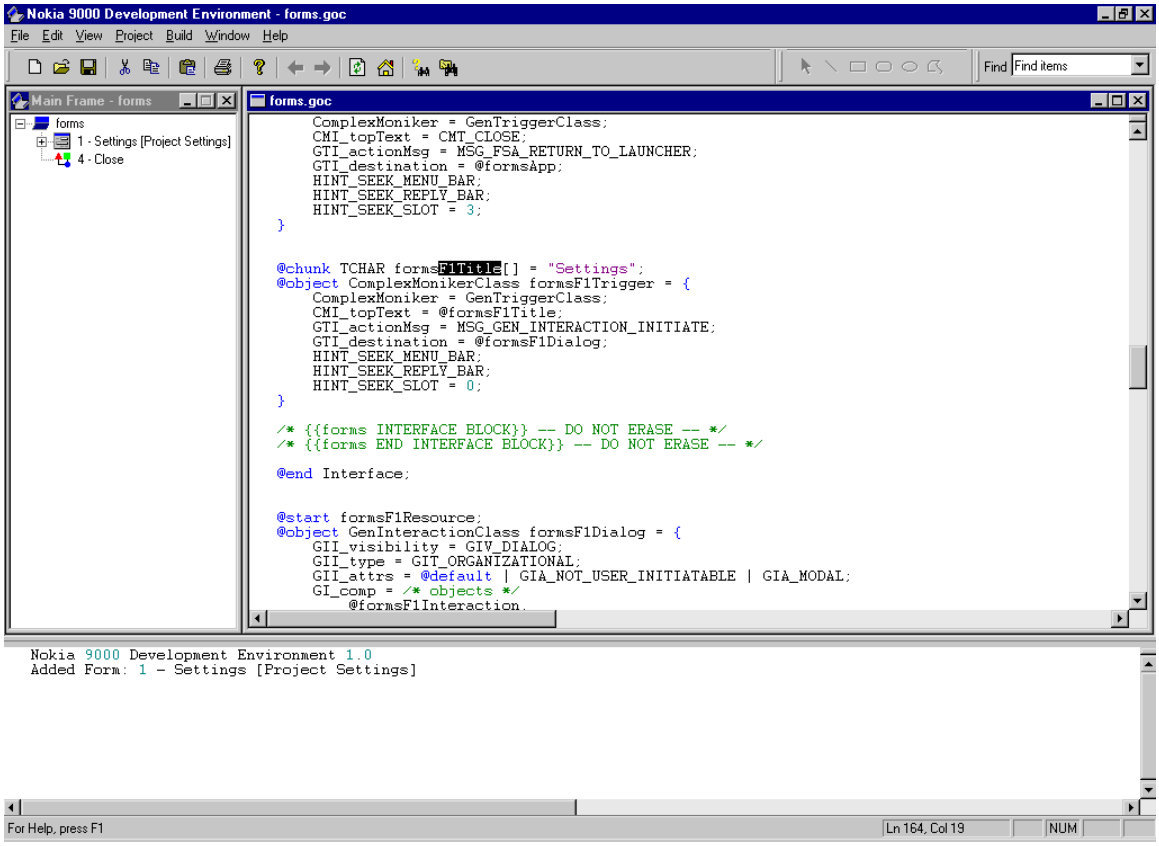
Step 4

By clicking on “OK” in the Add New Form dialog, you are adding a new set of form and trigger code to the files in your project. Immediately, you can see the change in the project tree view, which displays the new “Project Settings” form.



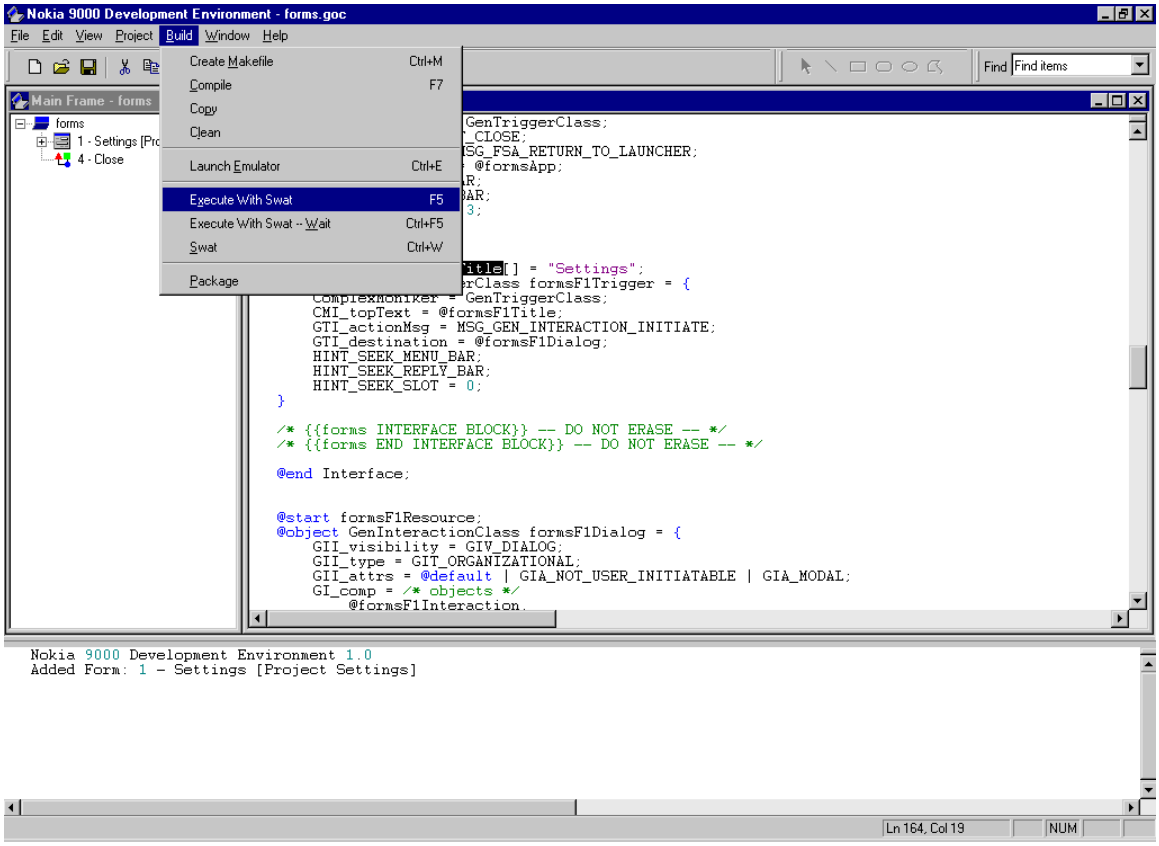
Note that you can resize the windows within the N9kDE workspace by grabbing the divider bar and dragging one direction or another. The cursor changes to a bi-directional arrow when you can resize the internal window.

Next, expand the tree view for the new form and double-click on the new form (named “Project Settings”) to see the GOC file.

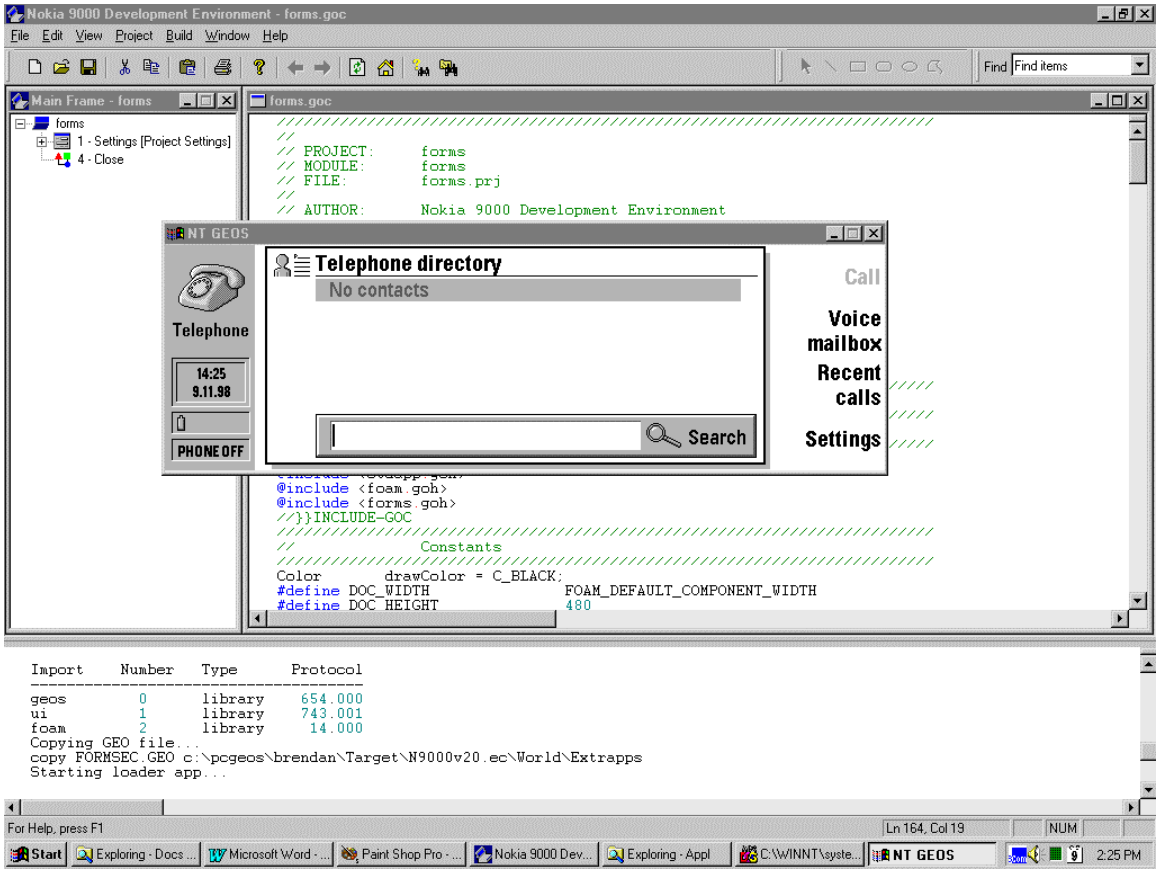


Notice that the SDK automatically takes you to the line in the GOC file which deals with the new form. Internally, the SDK keeps up with forms using a hierarchical identifier such as F1 for a level 1 form.

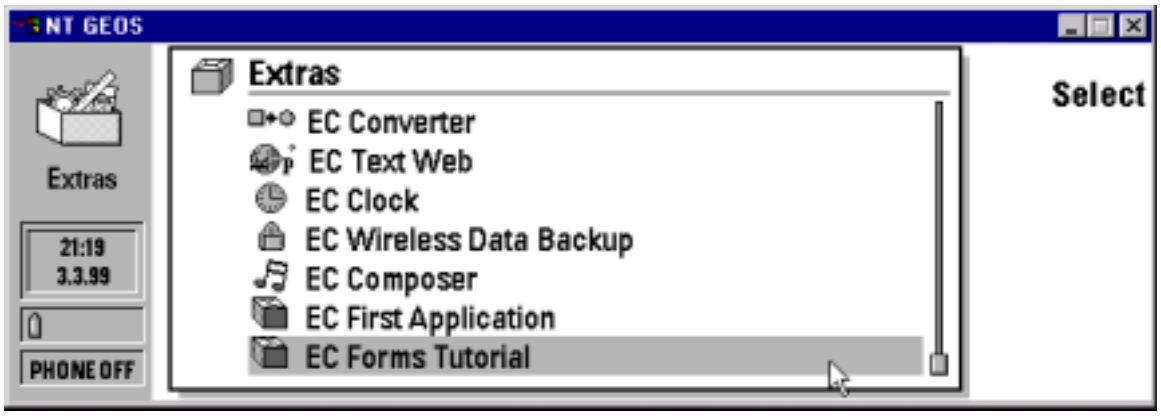
Next, compile and execute the project. Choose Build->Execute with Swat from the menubar.



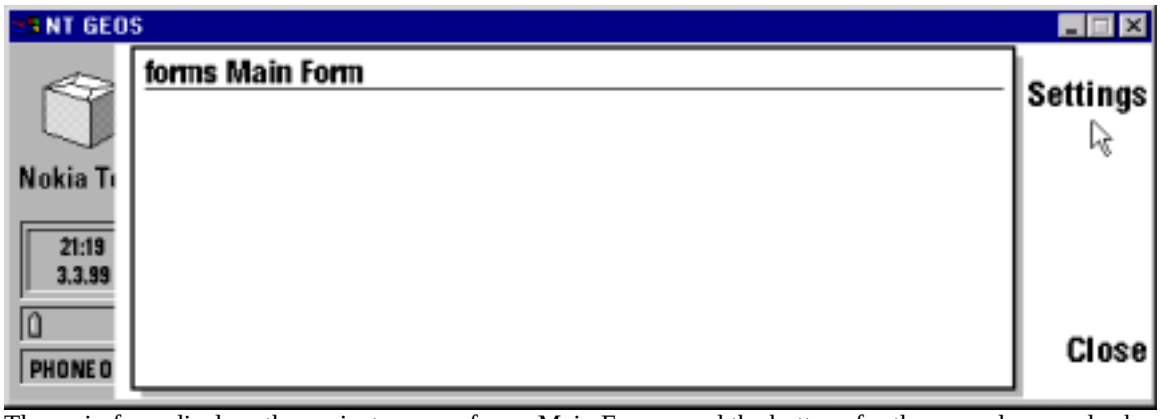
You may find it helpful to increase the size of the output window to view the makefile and compiler reports.



Press <CTRL><F12> to switch to the Extras applications. Then navigate to your application.

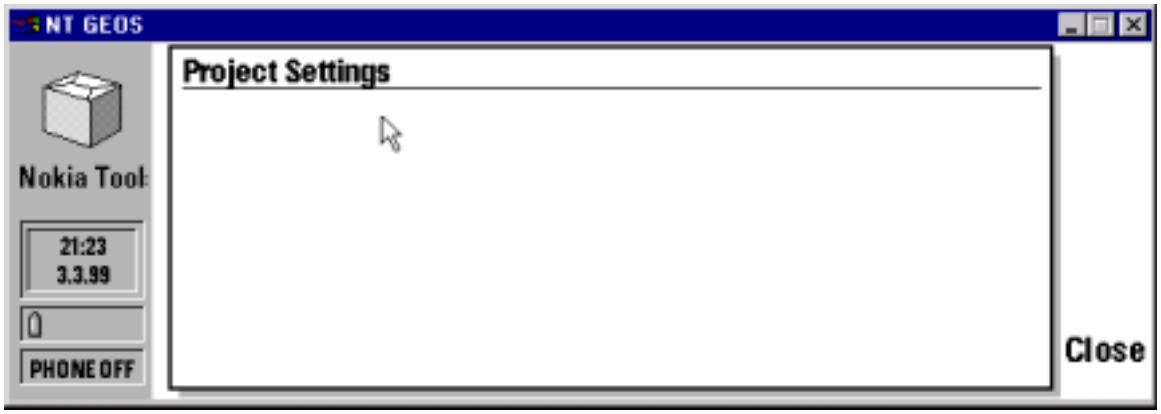


Launch your application (Forms Tutorial) by pressing <RETURN> or pressing <F1> to select.



The main form displays the project name – forms Main Form – and the buttons for the procedures and sub-forms. Notice that the newly created “Settings” button in position 1 appears on the emulator.

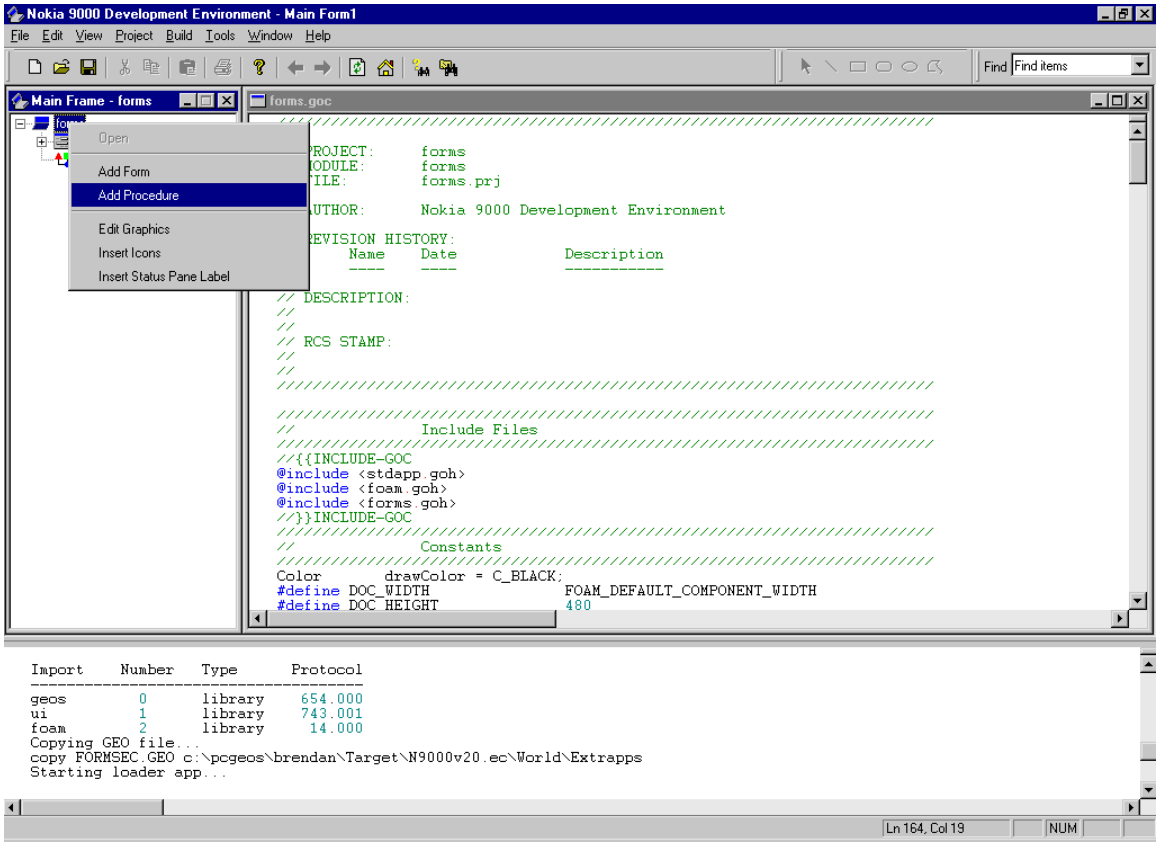
Choose the “Settings” button by pressing <F1>.



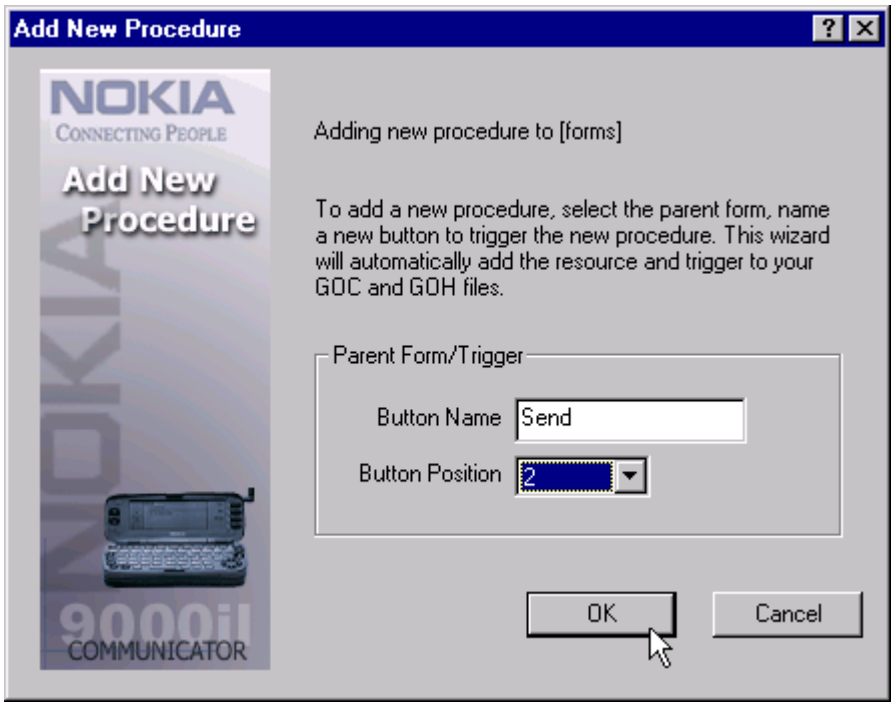
The new “Project Settings” form is displayed. This is one of the best features of the N9kDE, since it permits rapid prototyping of large applications.

Choose <F4> to close the “Project Settings” form. Choose <F4> again to close your “Forms Tutorial” applicaton. Then exit the emulator by pressing the dismiss icon in the upper right corner.

The next step involves adding a procedure (trigger) to the project. Although you can add both a form and a procedure from the Project… menubar option, use the popup menu feature of the tree view display to add this procedure. Position your mouse over the form to which you want to add a procedure (trigger) and click the right mouse key.

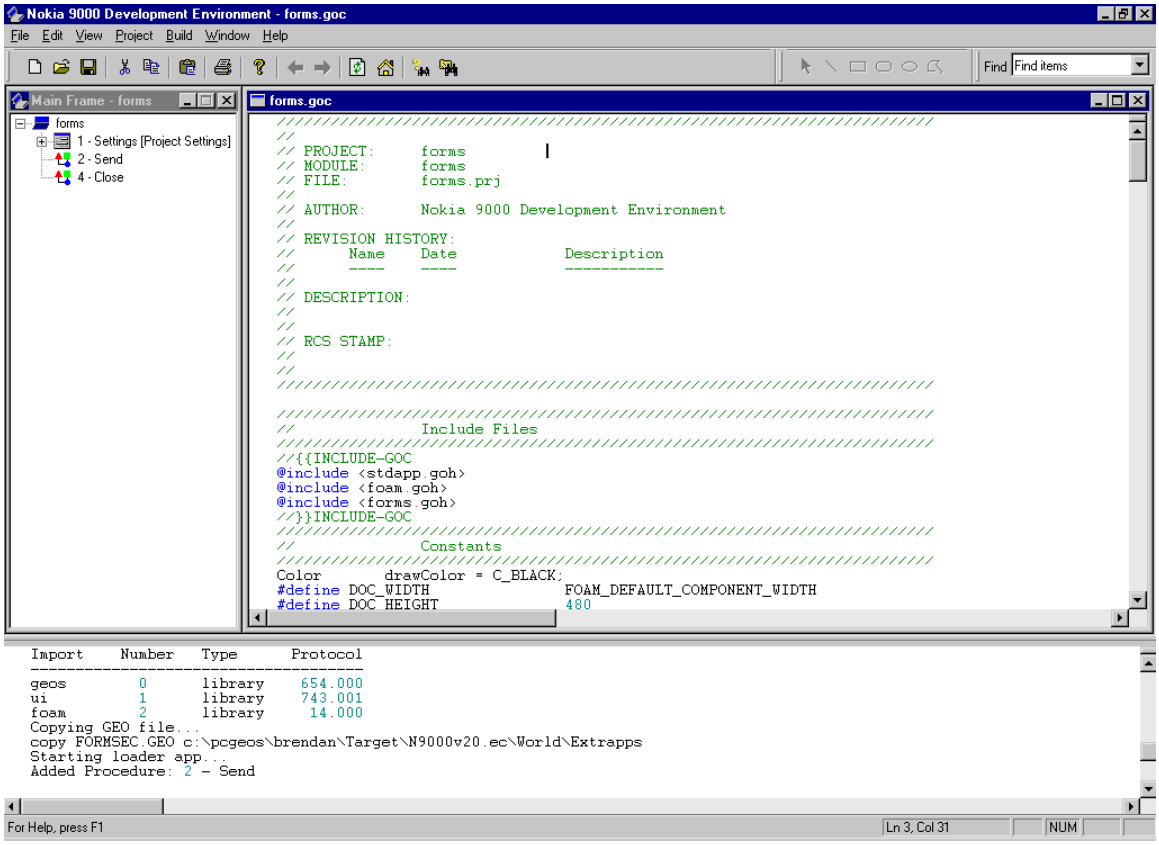


Choose the Add Procedure option. Enter “Send” as the button name. Enter “2” as the button position.



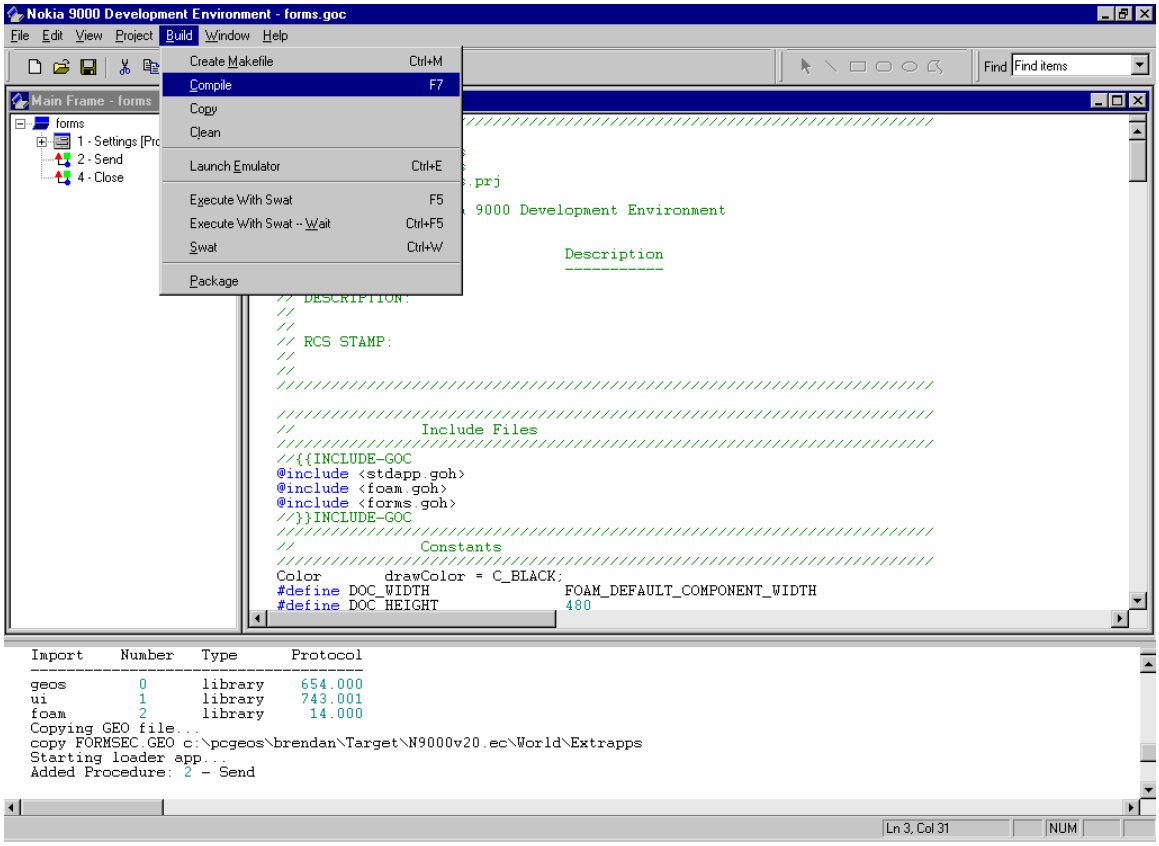
Click OK to create the new button.

The new button is added to the project tree view.

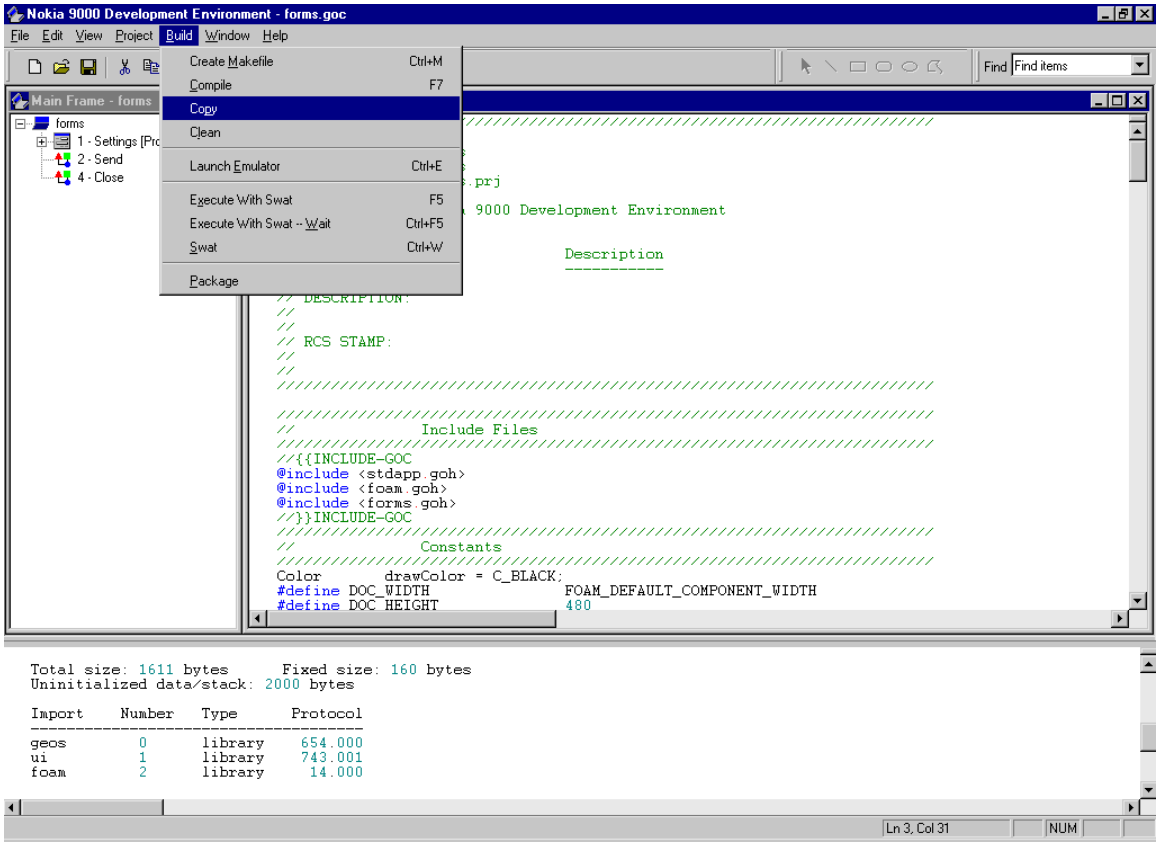


Notice that the output window displays the addition of procedure “Send” in button position “2”.

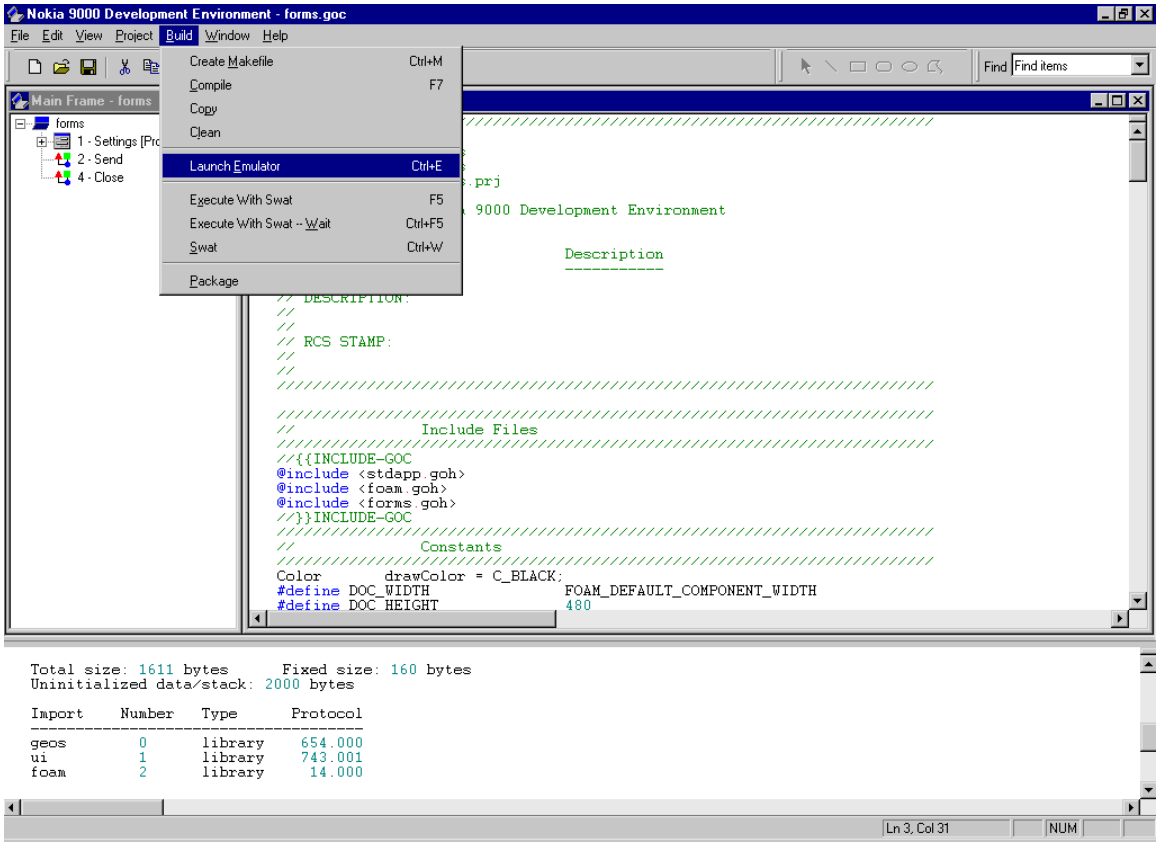
Since the basic project did not change, only a re-compile is required. Choose Compile from the Build… menu, or press <F7>.



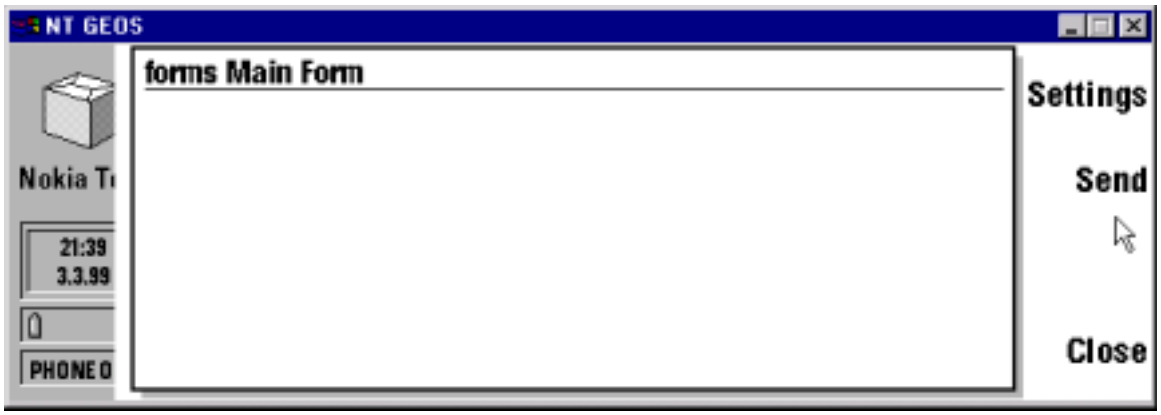
Next, choose Copy from the Build menu to copy the GEO file from the project directory (D:\pcgeos\Administrator\Appl\forms\FORMSEC.GEO) to the target directory. The default target directory is “EC” – error correcting.



Next, launch the emulator.

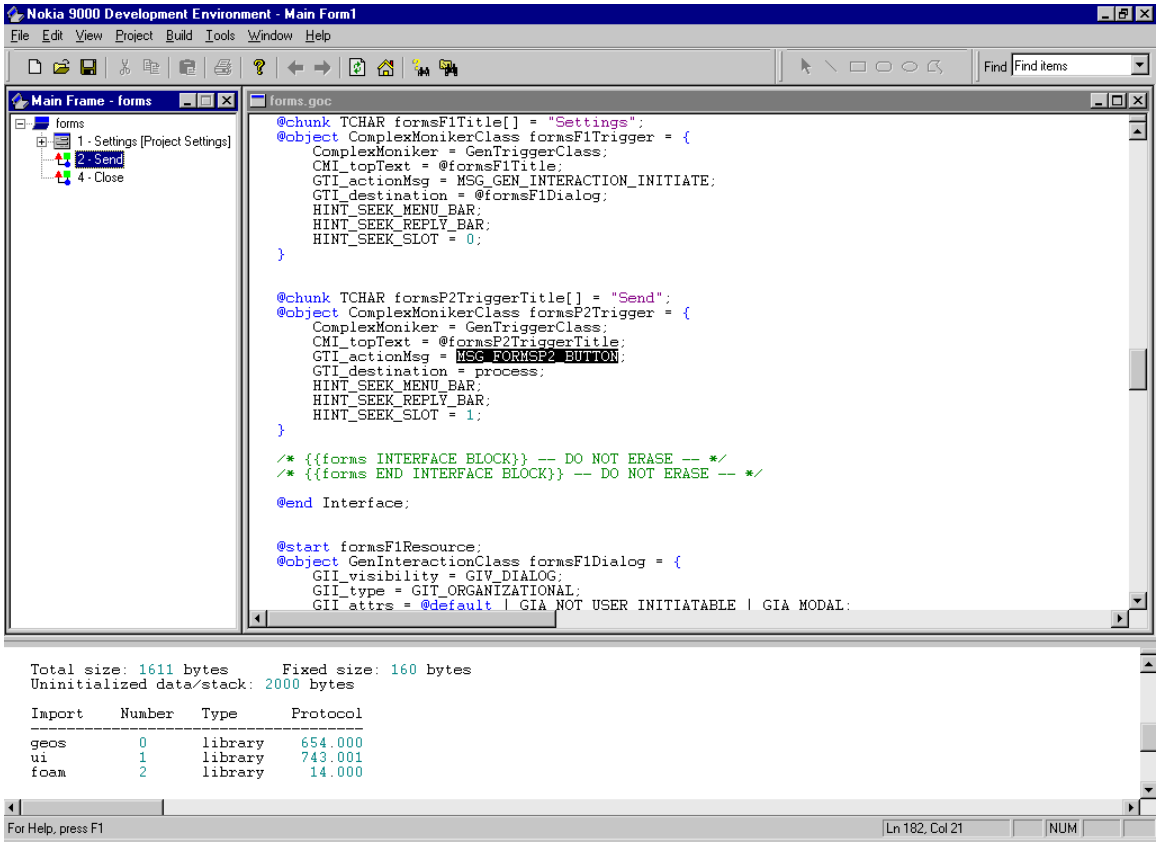


Go the Extras applications by pressing <CTRL><F12>. Locate the forms Tutorial application and press <RETURN> or <F1> to select. The new “Send” button appears in position 2 of the main form.



To activate the send button, press <F2>, which is the emulator command for button 2. In this case, no noticeable change in state occurs since there is no code associated with the send procedure.

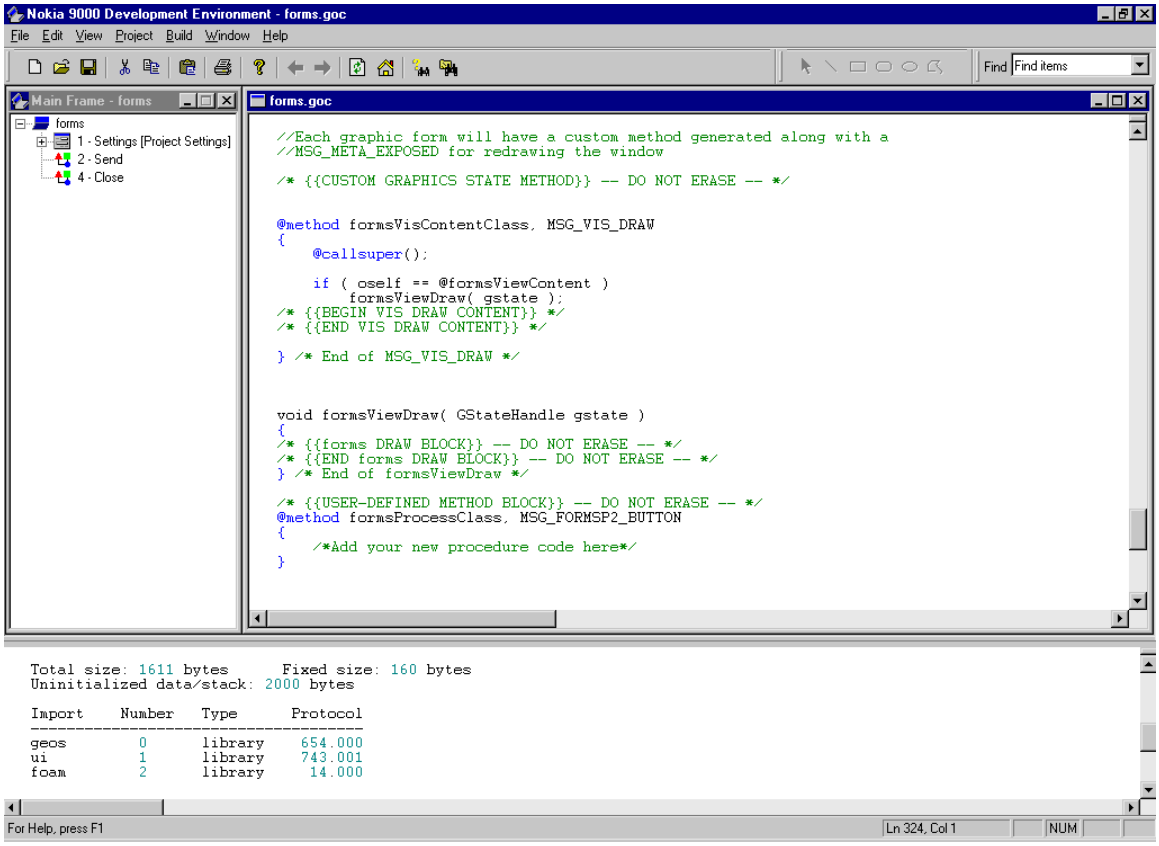
To examine the code for button 2, double click on the “Send” button in the tree view. The N9kDE positions the GOC editor at the line in the file for procedure 2 of the main form.



To find additional references to the “Send” procedure, use the Find option of the editor. Click <CTRL>F to launch the find dialog.

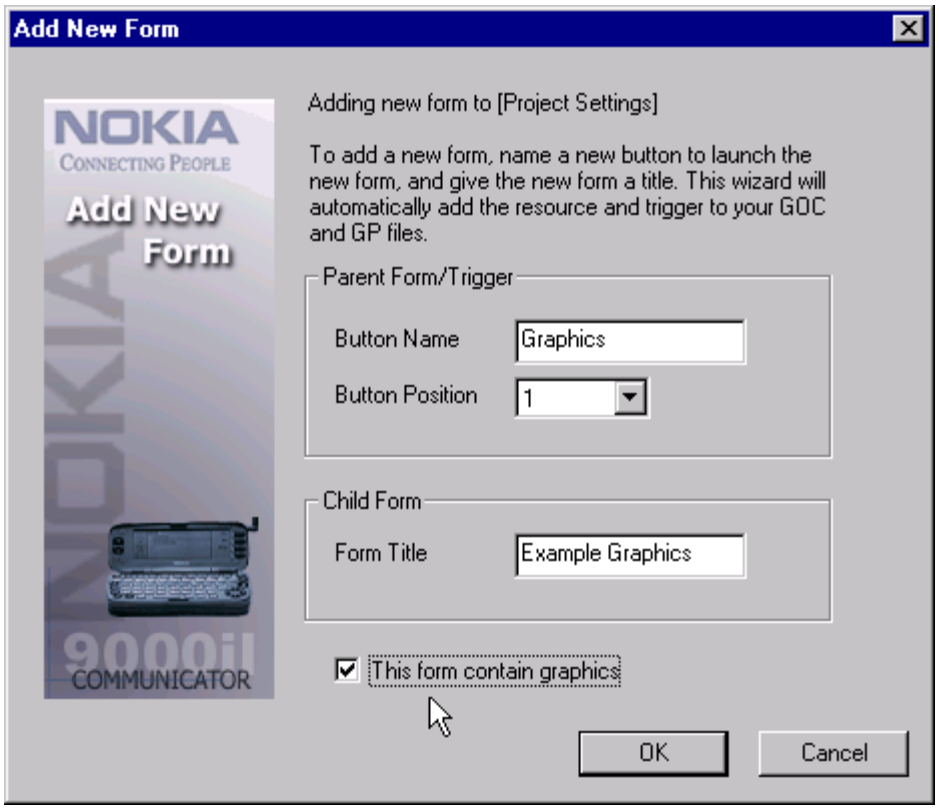
Use the <F3> key to perform a find next command.

The default procedure created by the SDK does not include any special logic. It simply creates a skeleton procedure for the developer to modify.

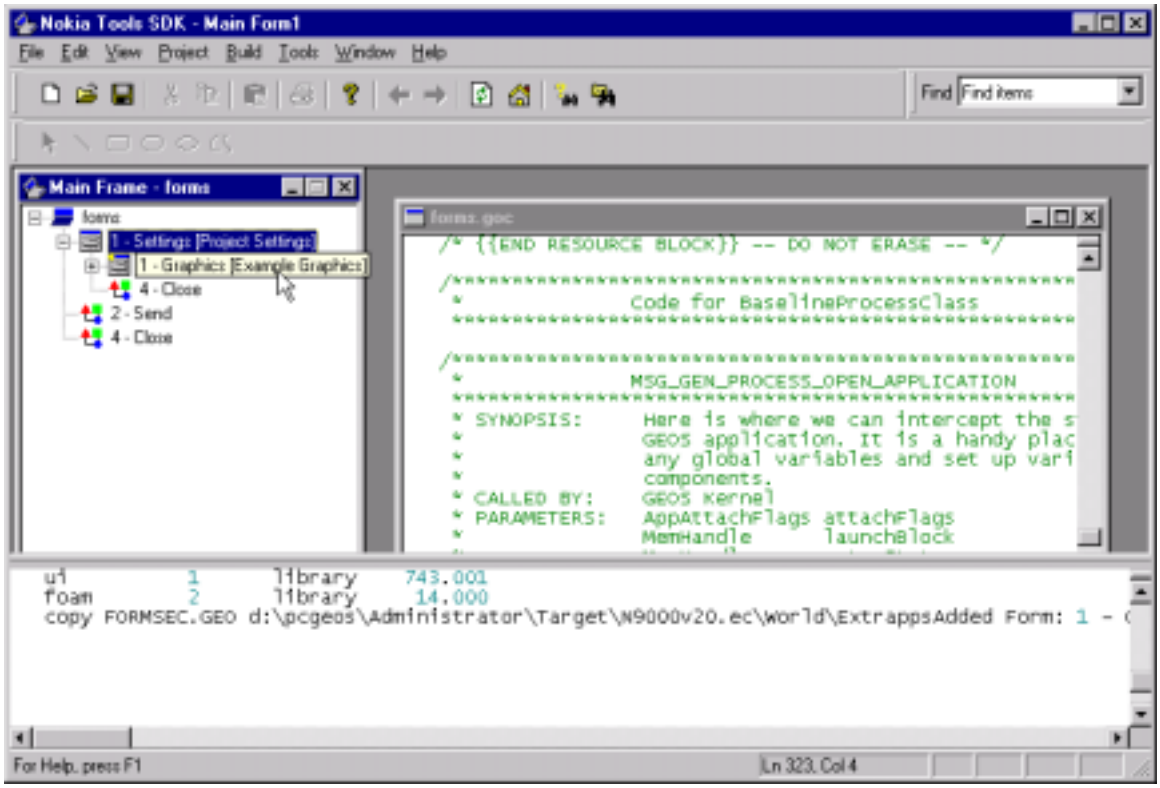


Next, we will insert a graphic form into the project. From the Project menu, choose the “Add form” option again. Not that the form is added to the currently selected form in the tree view. Select the “Settings” form prior to attempting to add a form to the project.

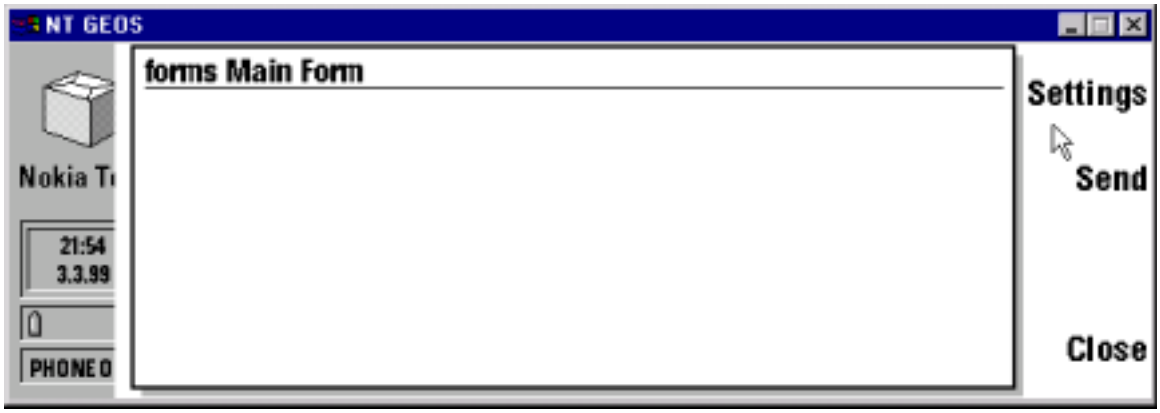
The Add New Form dialog displays “Adding new form to [Project Settings]”, which is correct. We enter “Graphics” for the button name, “1” for the button position, and “Example Graphics” for the form title. Make sure that you check “This form contains graphics” to enable the graphics option. You must choose the graphics checkbox to add lines, rectangles, and circle shapes to the form display.



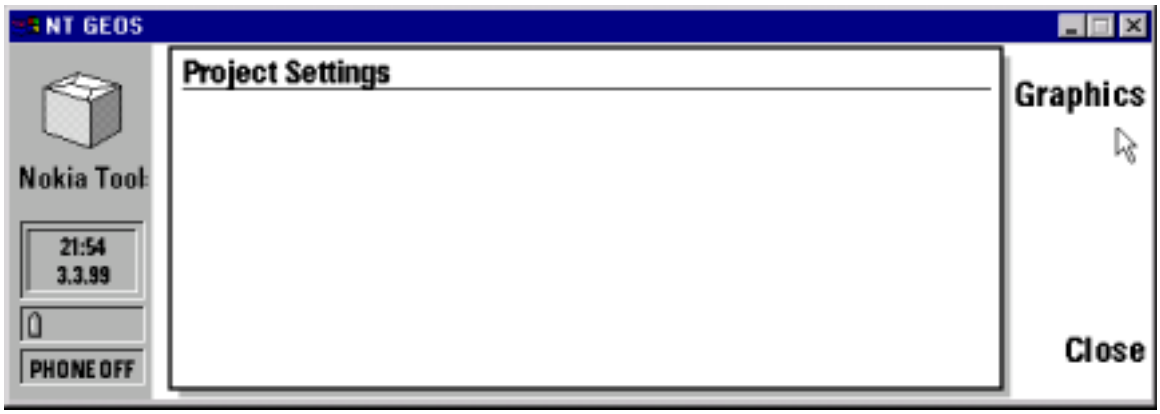
The new graphics form appears under the Settings form in the tree view.



Recompile and execute the emulator. Choose <CTRL><F12> to go to the Extras applications.

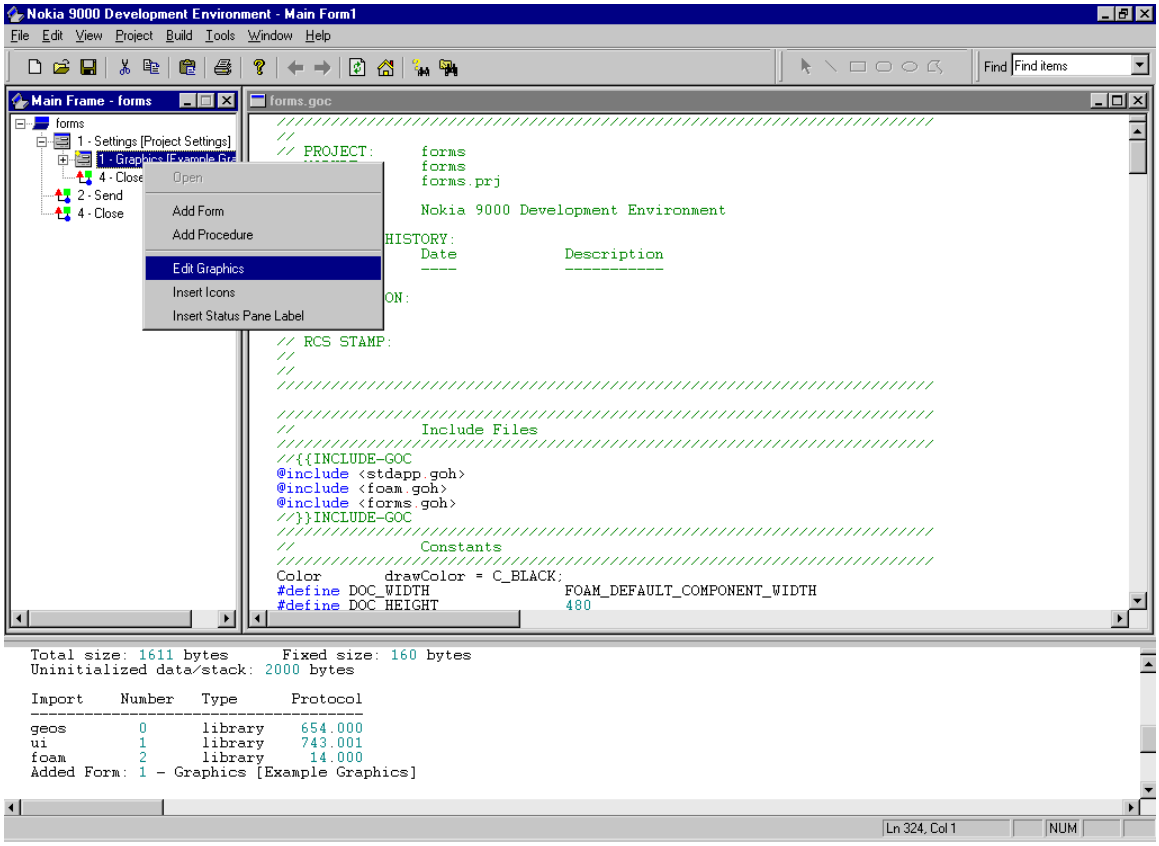


Now choose Settings <F1> from the main form. The Graphics button, which launches the Example graphics form, appears.

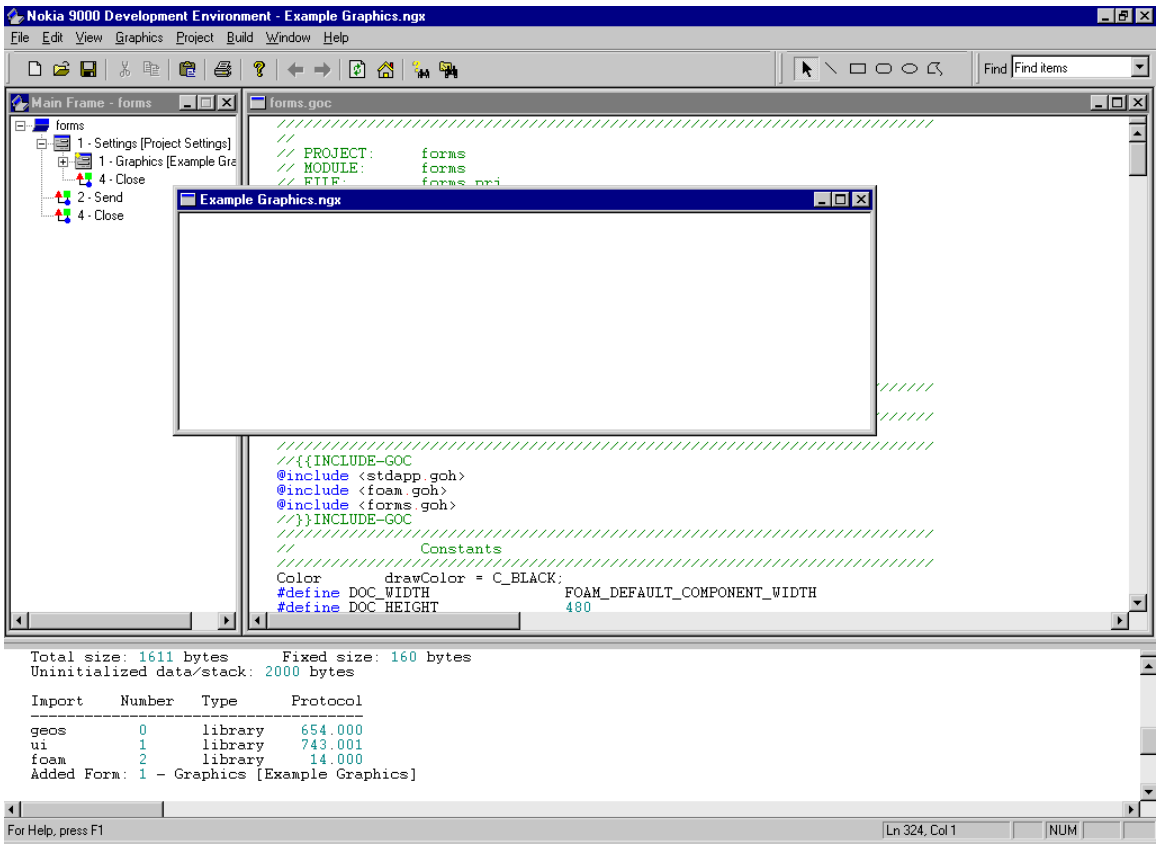


Choose <F1> to launch the graphics form. At this point, the graphics form does not include any drawing. Close the application and emulator so that next, we can add drawings to the graphics form.

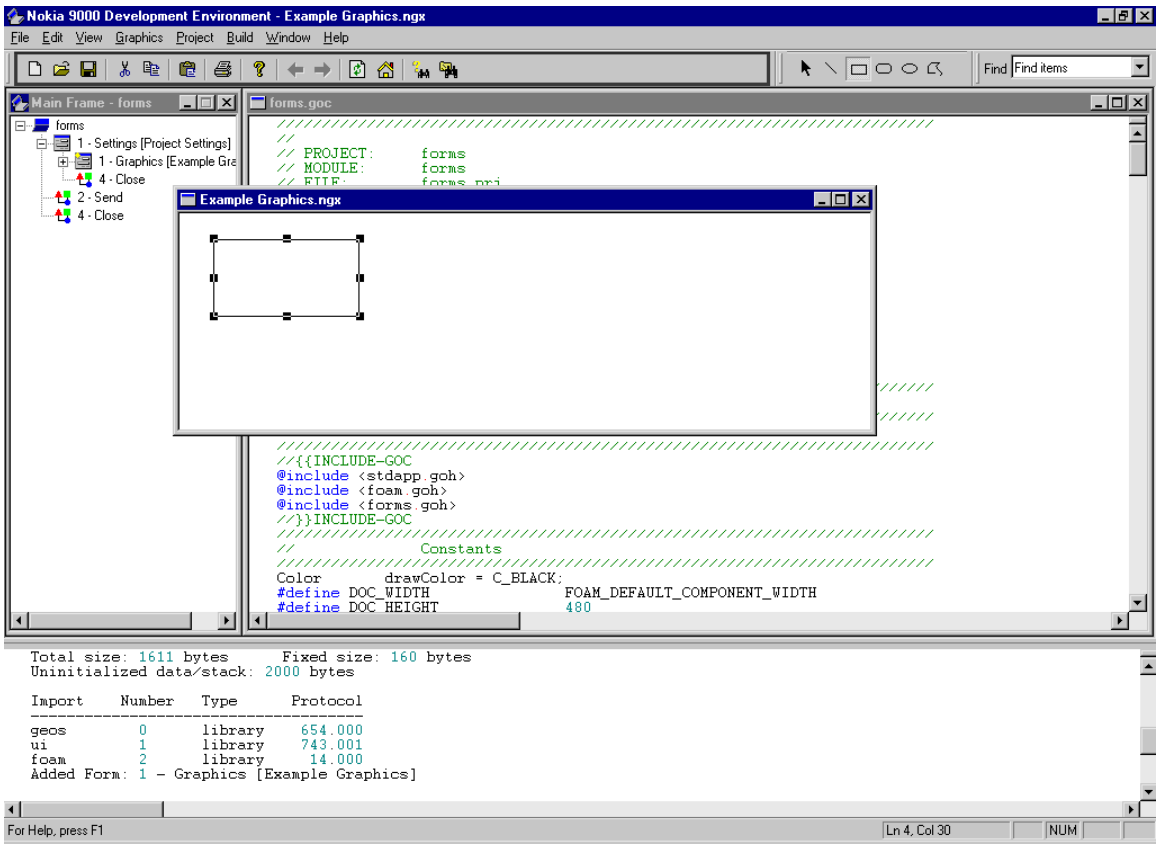
Choose “Example Graphics” from the project tree view. Right click the mouse to see the menu options. Choose “Edit Graphics”



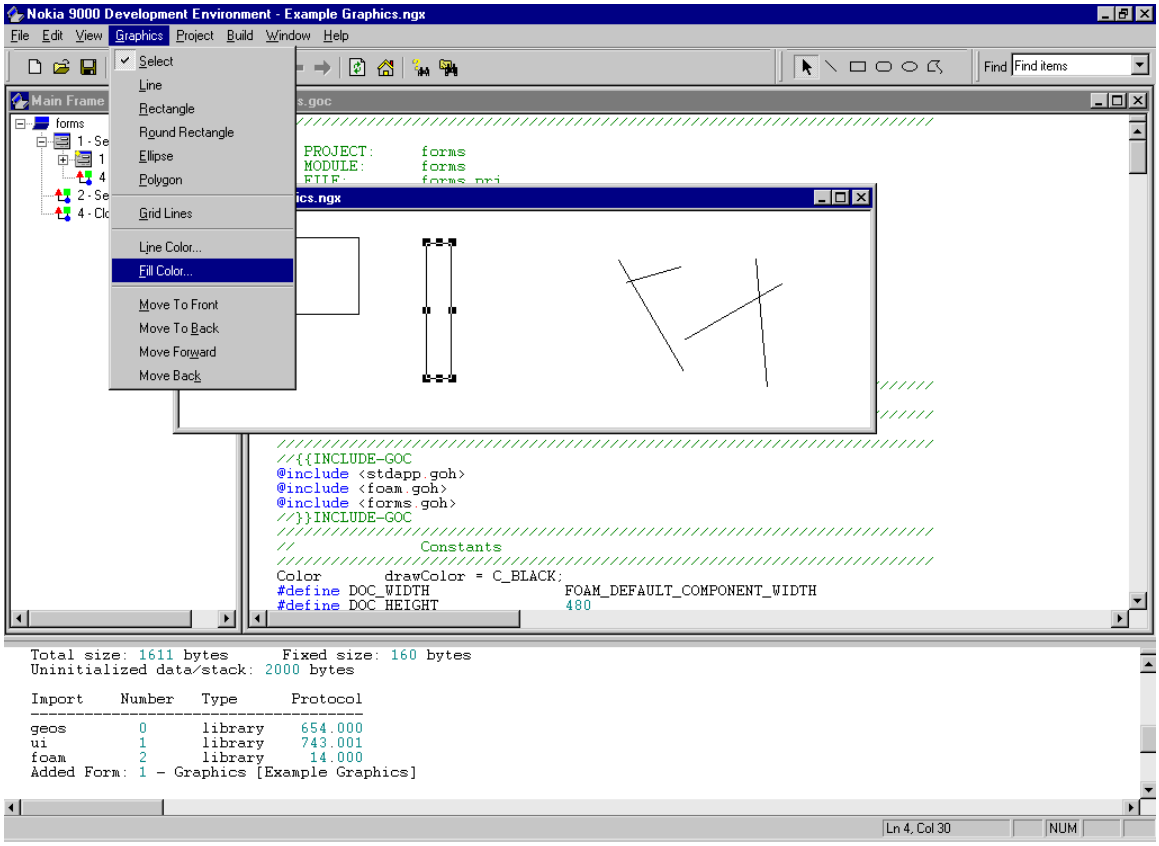
The empty graphics screen appears.



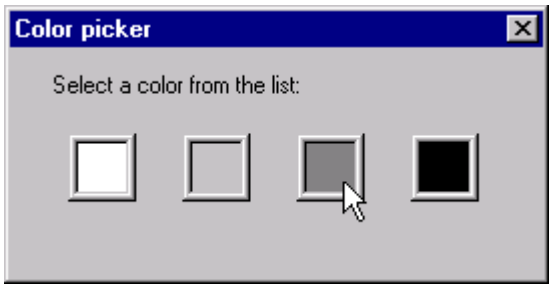
Select the rectangle drawing tool  from the toolbar. Drag the mouse from the top left coordinate of the rectangle to the lower right corner.



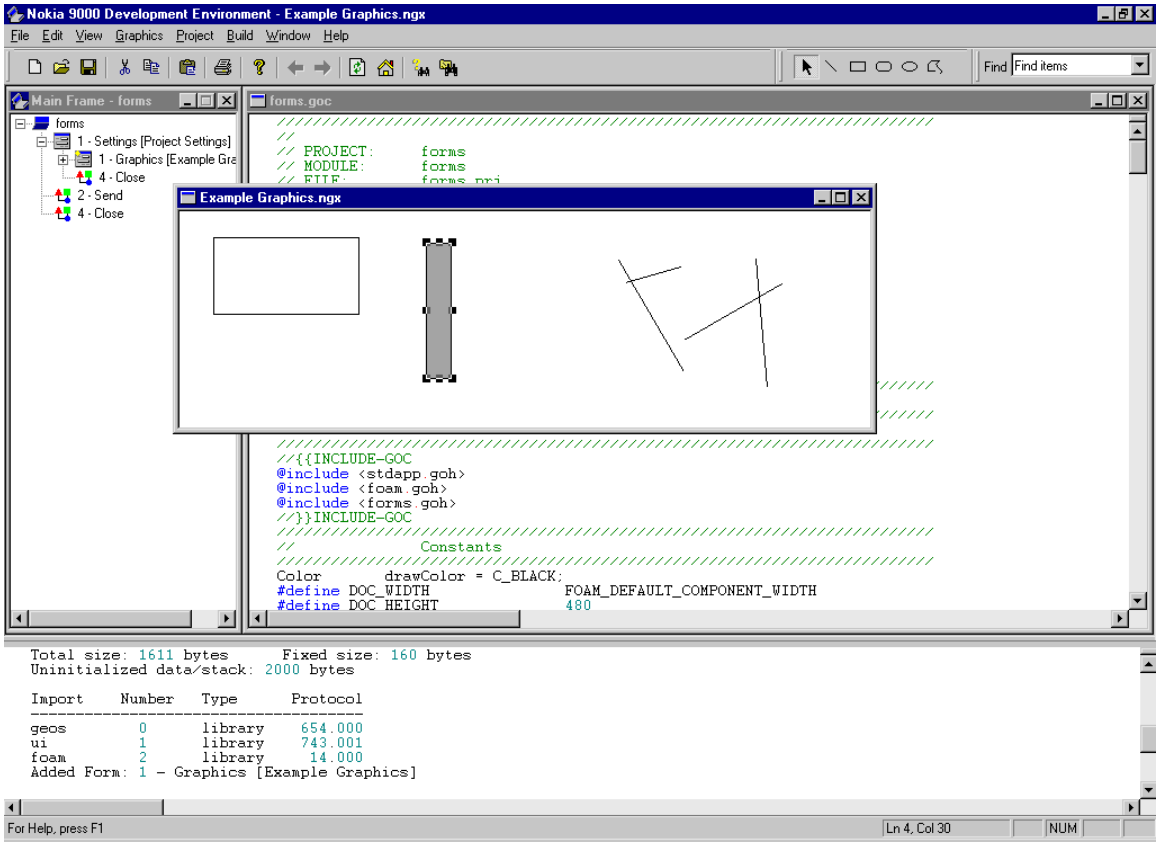
Draw a few more rectangles and lines. Next select a rectangle and choose the Graphics option from the menubar.



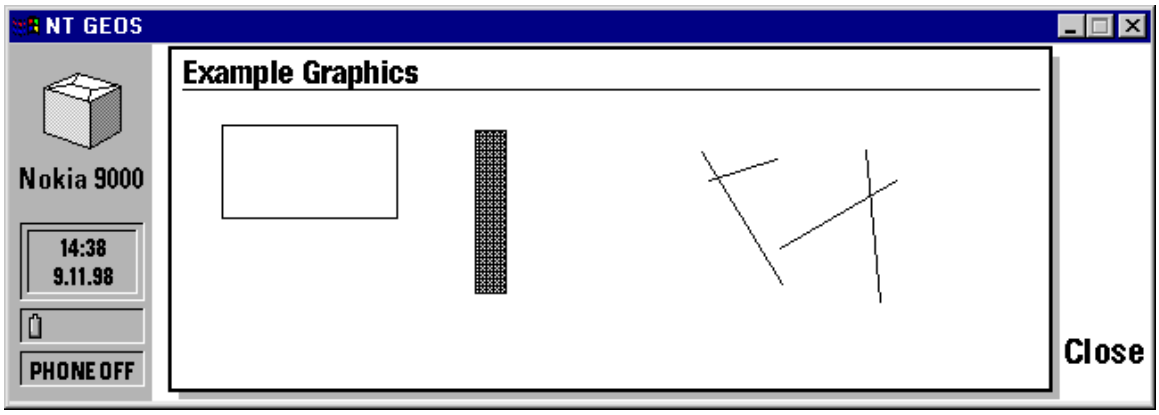
The fill color will change the fill color (GrState) to one of the four grayshade levels supported by the N9000.



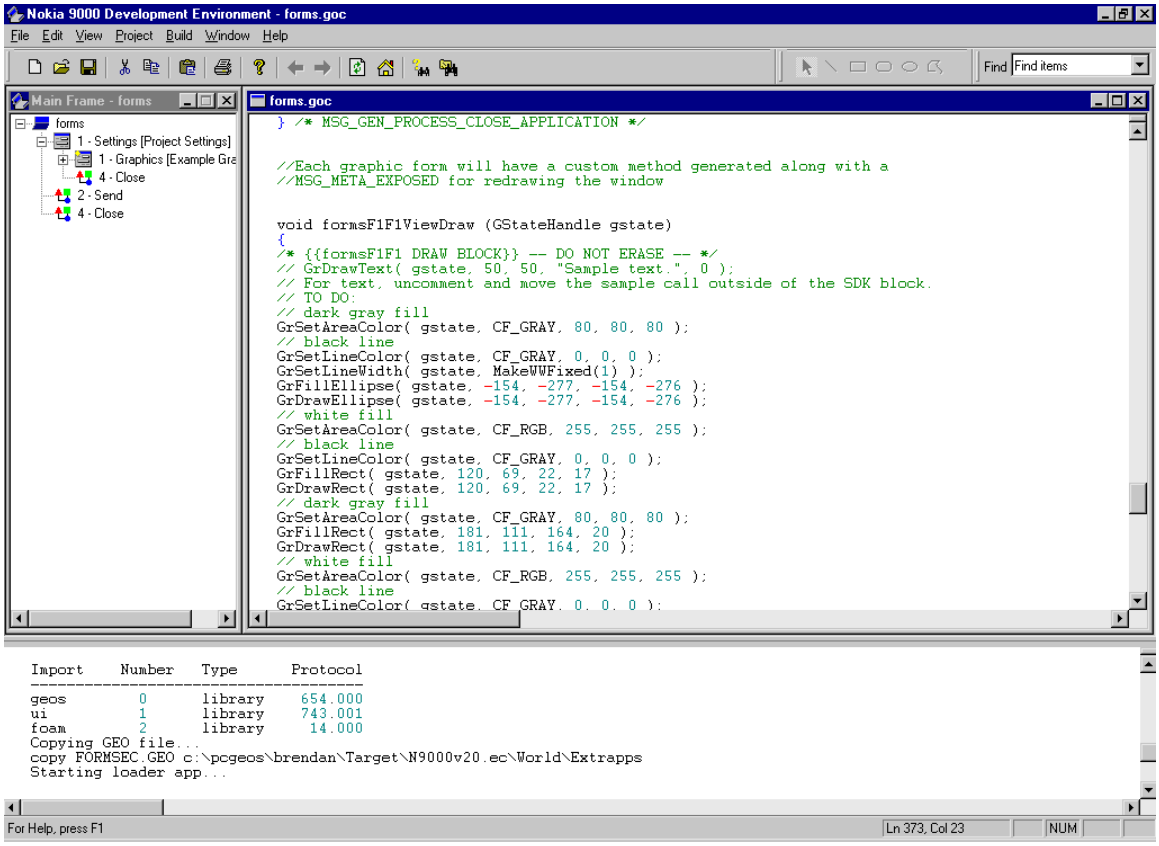
Changing the fill color changes the graphics display.



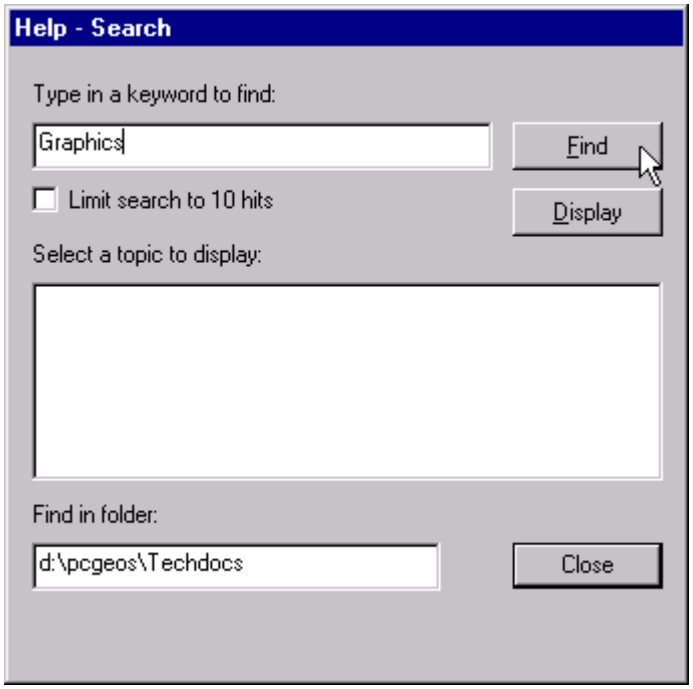
Re-run the project in the emulator <F5> to see how the graphics are displayed.



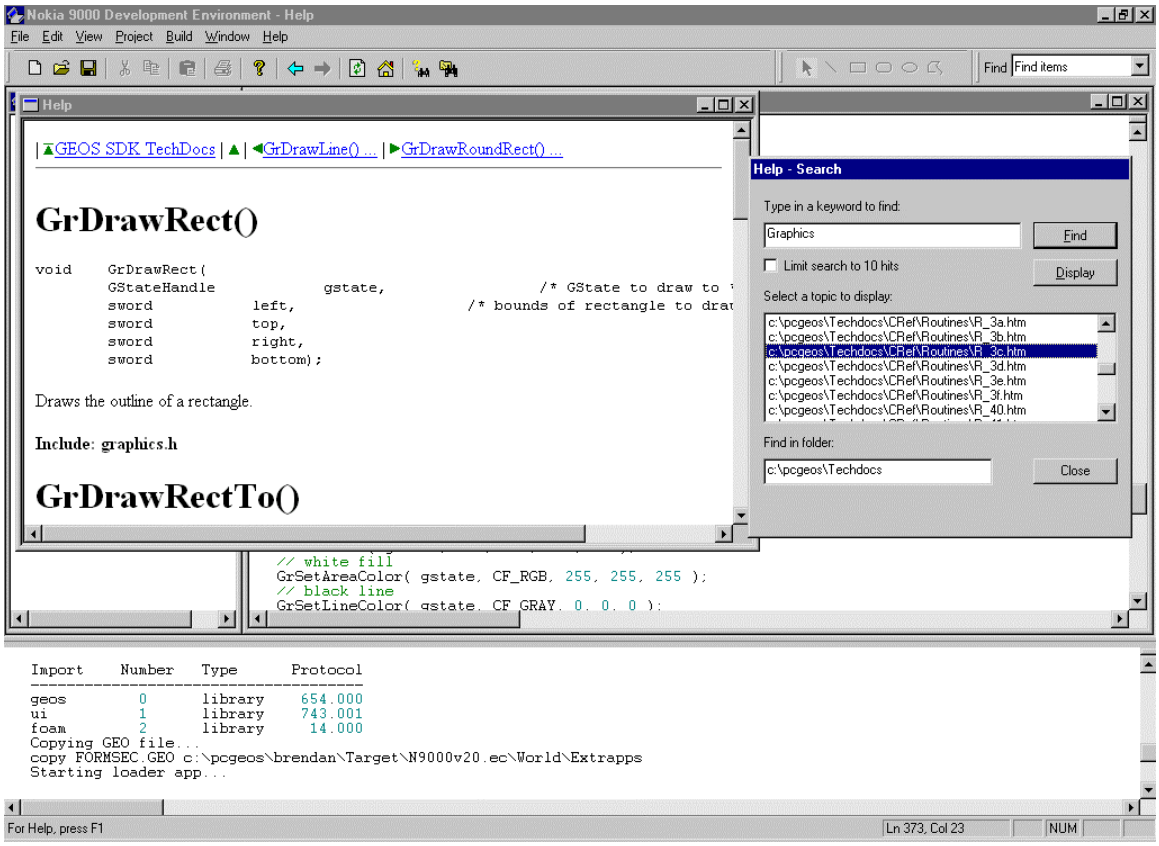
Exit the emulator and examine the source code associated with the graphics. Simply double click on the graphics form in the tree view. Find the keyword “Draw” to locate the drawing code in your graphics form.



To find out more about the graphics capability of the N9kDE, search  for “Graphics” in the online help, or choose Search from the Help menubar option.



Click “Find” to begin the file-by-file search for the topic of interest (e.g. Graphics). Searches through the Techdocs are case insensitive. Searches can take several moments to complete. You can interrupt a search by clicking Display at any time.



Close the Help Search dialog.

Save your project and exit the SDK. Congratulations! You have added graphics to a GEOS application through the N9kDE.

7 Tutorial 3 – Using the Toolbox

7.1 Objective

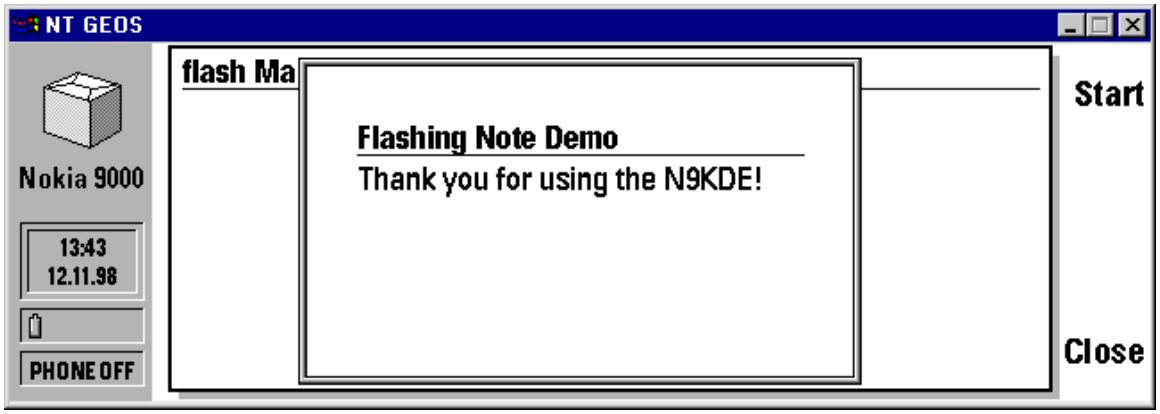
The objective of Tutorial 3 is to create a new project that includes the Flashing Note tool. The Flashing Note tool is a sample add-in tool for the SDK that can add code to your project which will display a custom flashing note. This tutorial demonstrates the Toolbox manager and more advanced application development.

The add-in tools are wizards that have the capability to quickly add new resources to your application. They can be built by third-parties using a specific API.

7.2 Using The Flashing Note Control

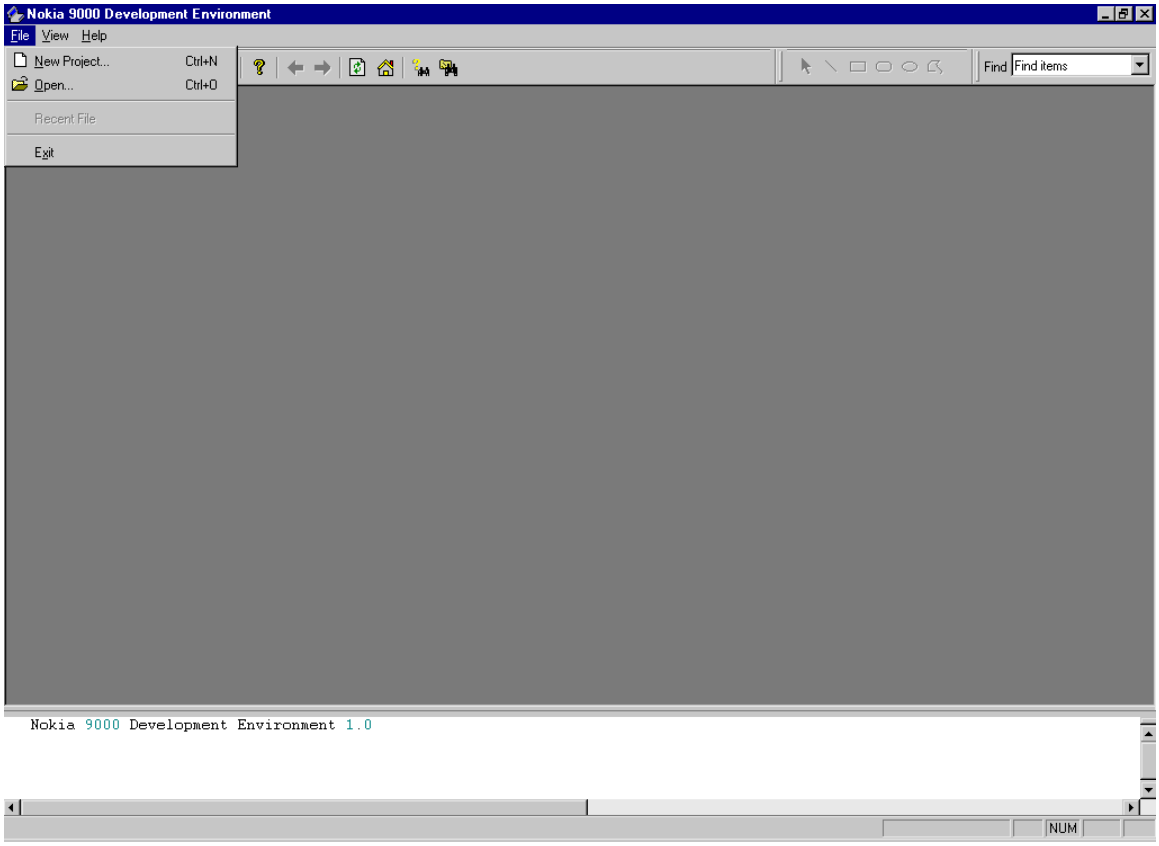
This tutorial shows you, the Nokia 9000/9110 Development Environment developer, how to incorporate the Flashing Note Control into your project. This tutorial assumes that you have followed all of the necessary installation procedures for the Nokia 9000/9110 Development Environment, Borland C++, and GEOS SDK.

The figure below shows an example of the flashing note that we will be developing in this tutorial. This is actually a screen snapshot of the emulator used within the Nokia 9000/9110 Development Environment.

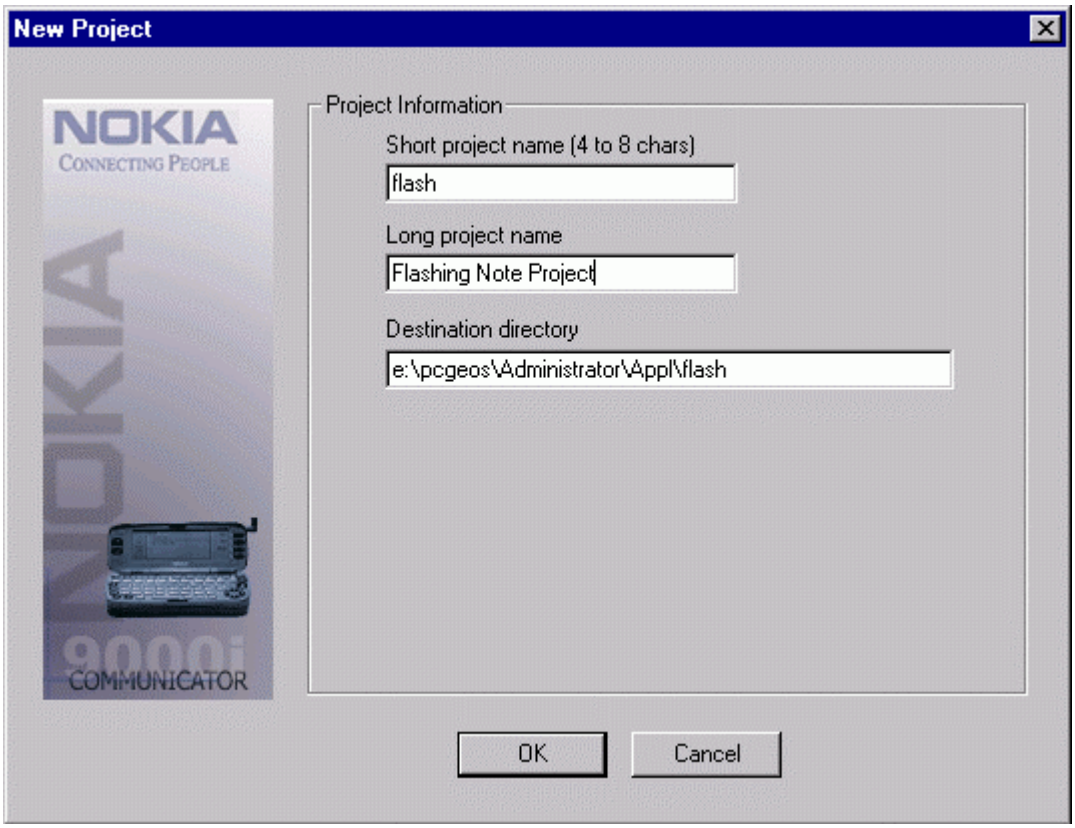


7.3 Procedure

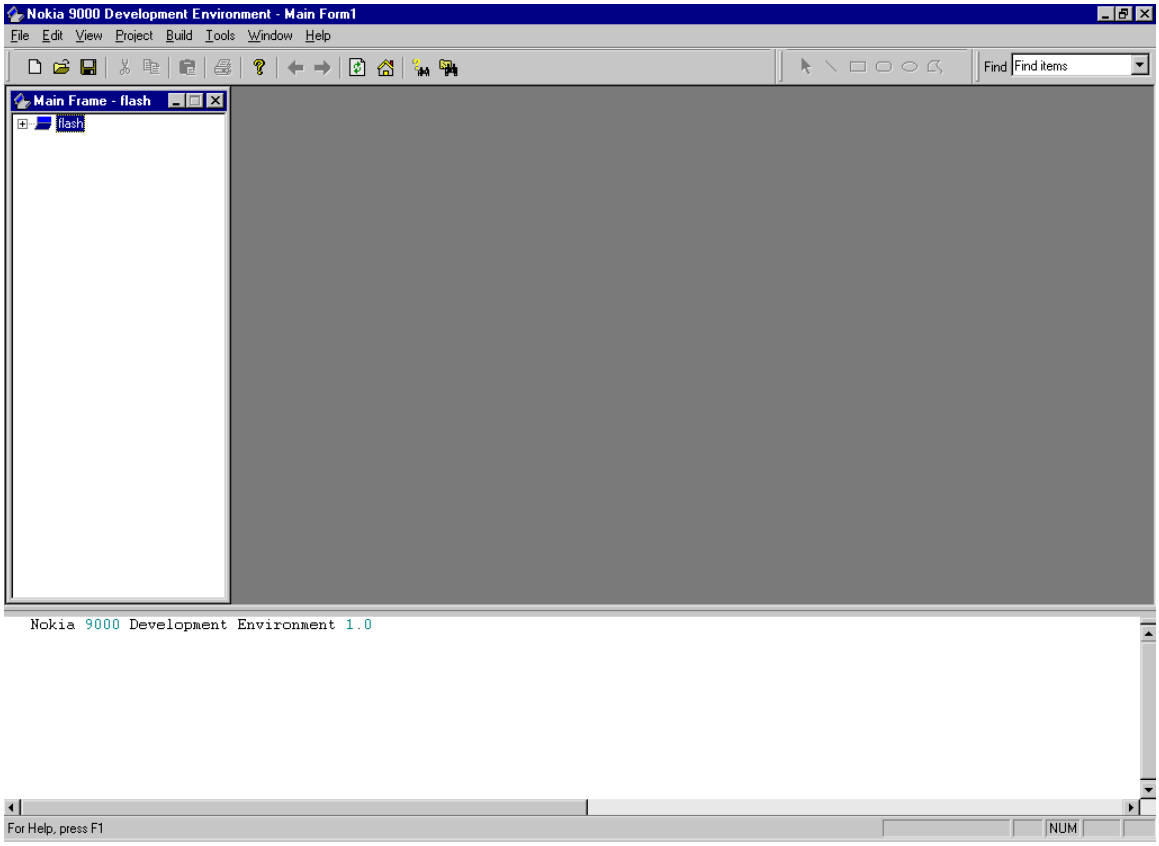
Once the SDK is started, you will select “New Project…” from the File menu.



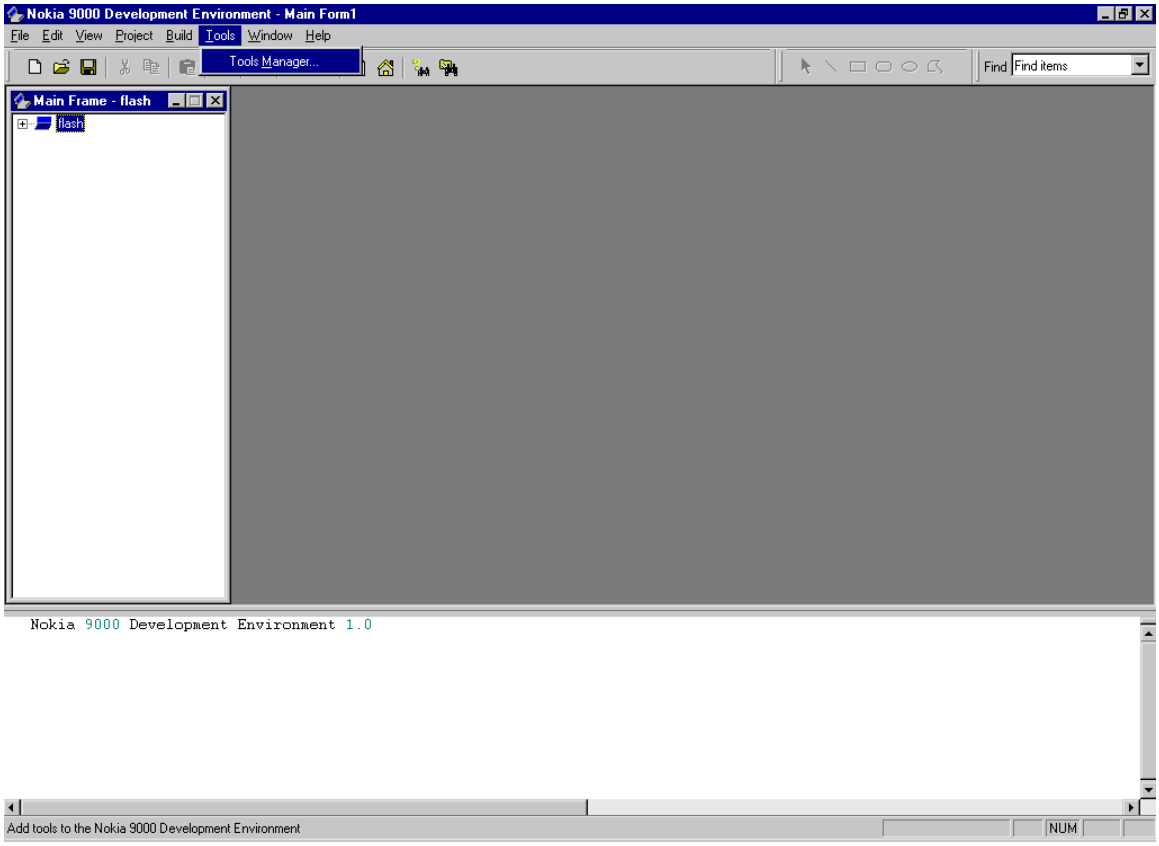
A New Project dialog will come up asking for project information including: project short name, project description, and the destination directory for your project files. The project name **must** be entered in lower case and must be between four and eight characters long.



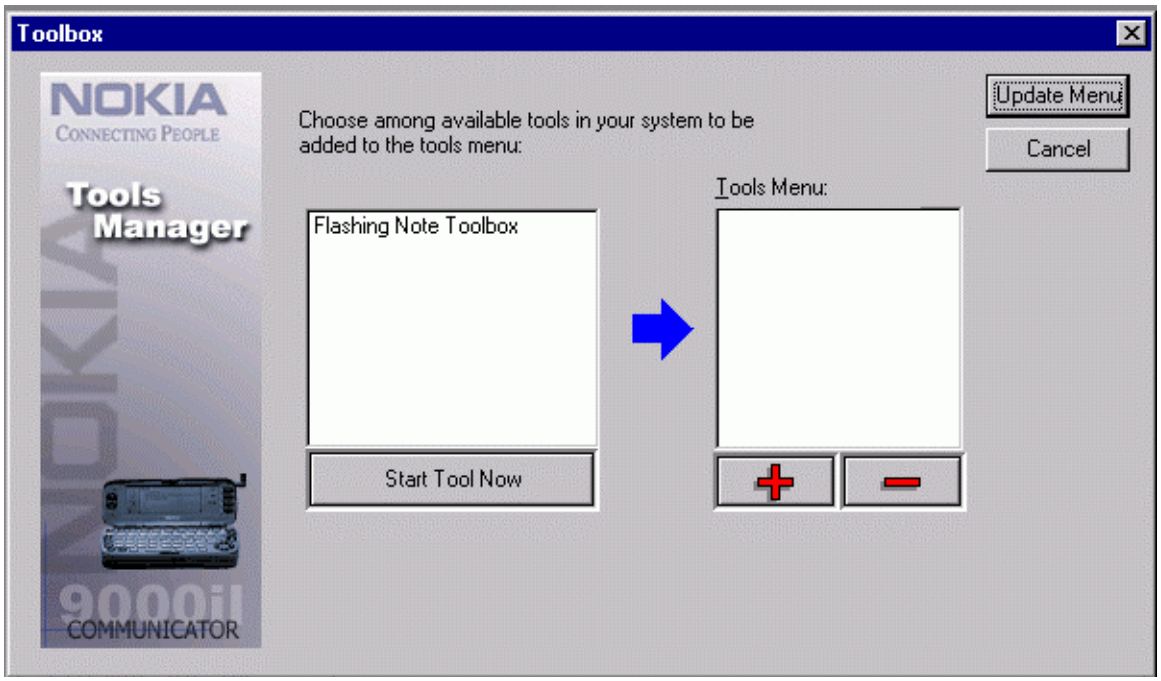
After selecting “OK”, the project files will be created and stored in the destination directory you provided above.



We will now add the “Flashing Note Control” to your project by selecting “Tools Manager” from the **Tools** menu.

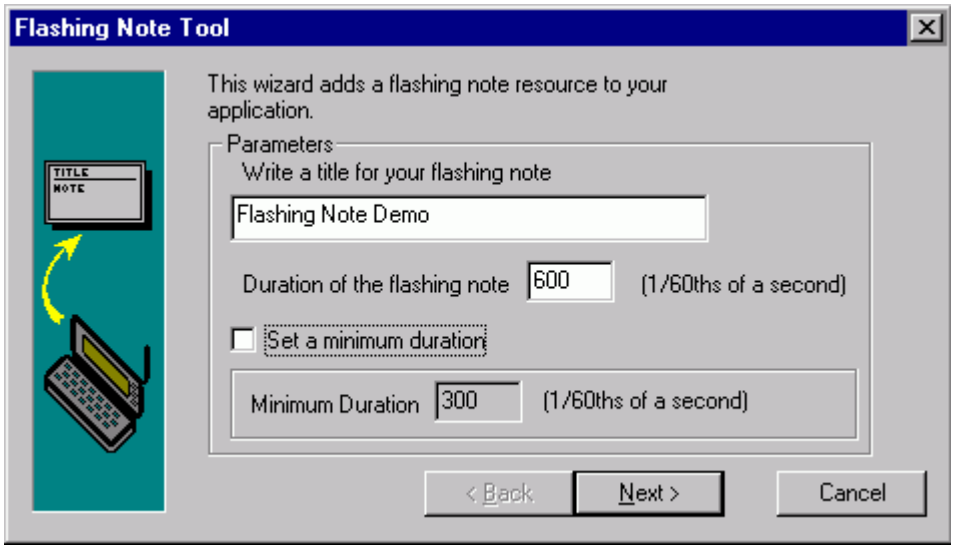


The Tools Manager will pop-up a dialog showing you what tools are available for you to add to your project. If you will be using the tool repeatedly, you may find it useful to add the tool to the Tools menu. Just highlight the tool you wish to add to your menu, press the “+” button, and press “Update Menu”. The tool will remain in the **Tools** menu unless you manually remove it.

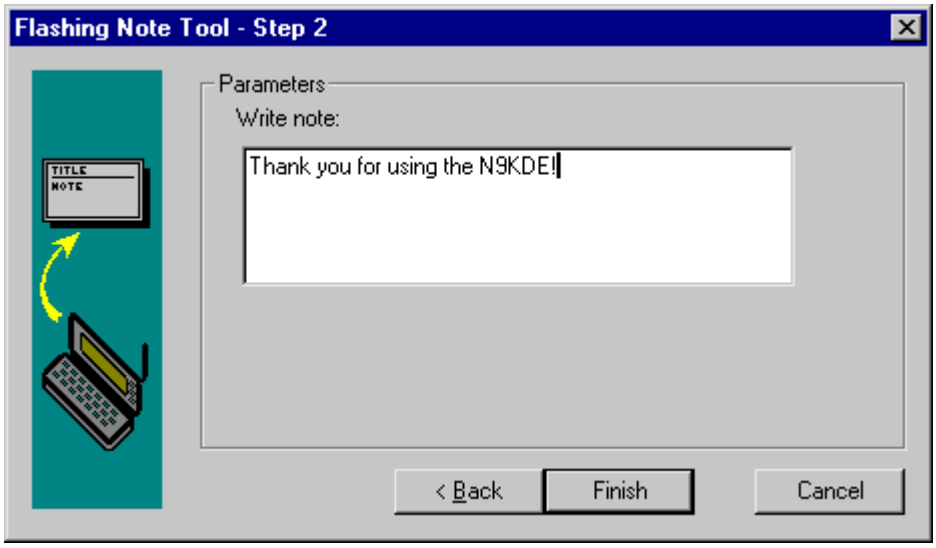


Select the “Flashing Note Toolbox” tool and press the Start Tool Now button. The Flashing Note Toolbox wizard will come up and guide you through the steps to create your flashing note.

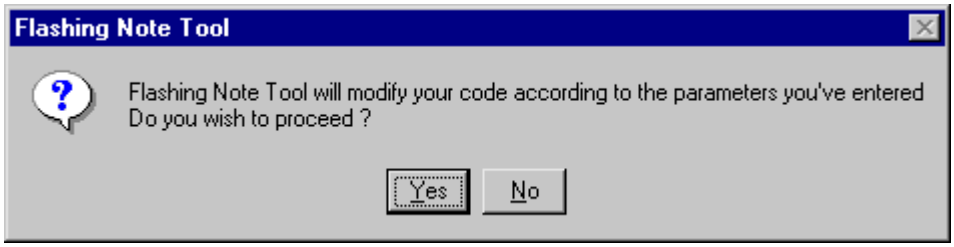
Provide a Title for your flashing note and press “Next>”.



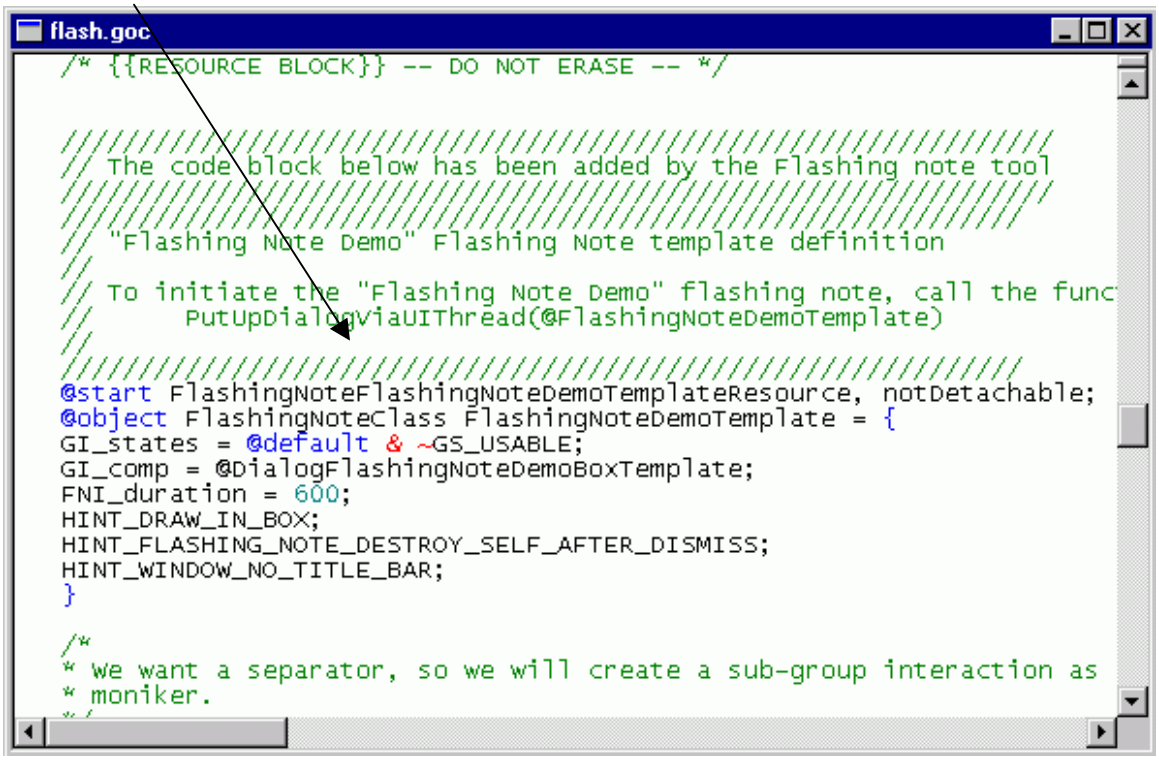
Enter your custom message that you wish to be displayed when your note is invoked and press the Finish button.



A message box will be displayed informing you that your project code will be modified to add the code for the flashing note based on the input above.

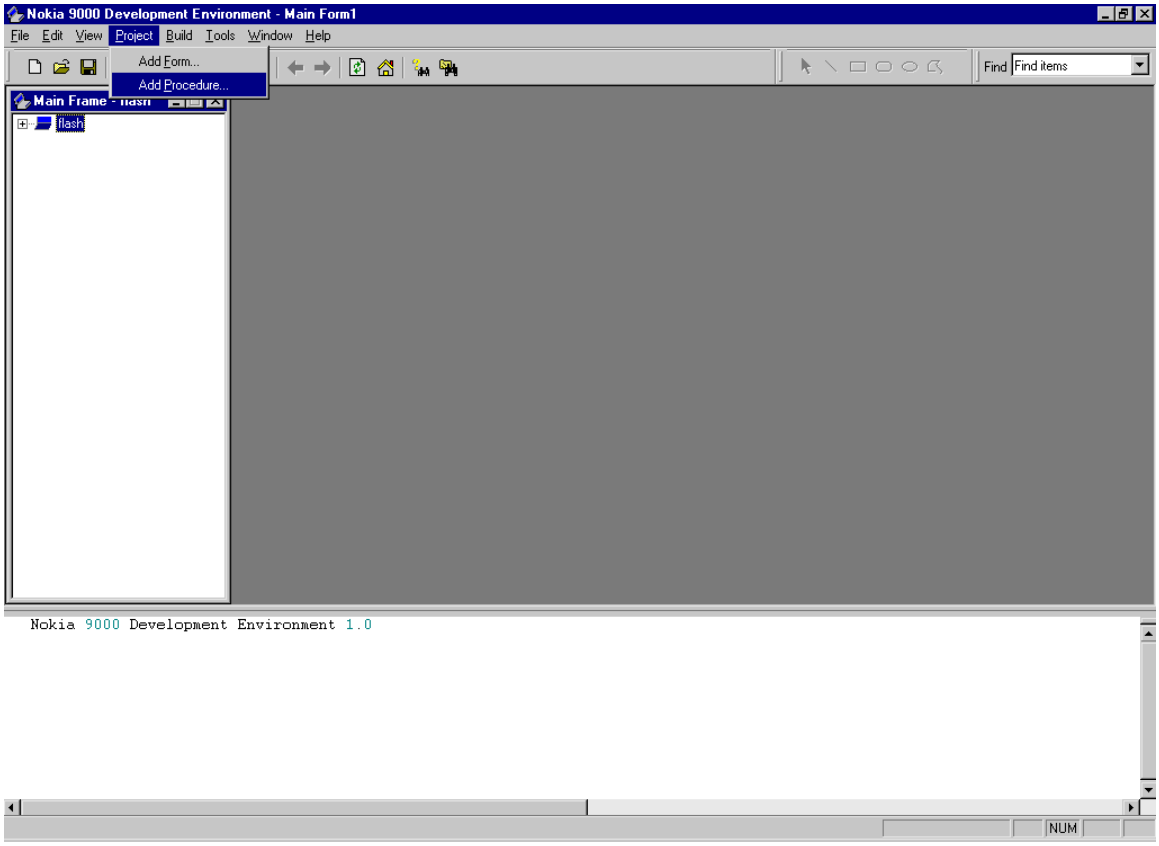


Selecting “Yes” will make the necessary changes to your code and return you to the tool manager dialog. Cancel the tool dialog and return to your project. This is a code snippet showing some of the code that is added to your project and the method that you need to invoke within your procedure.

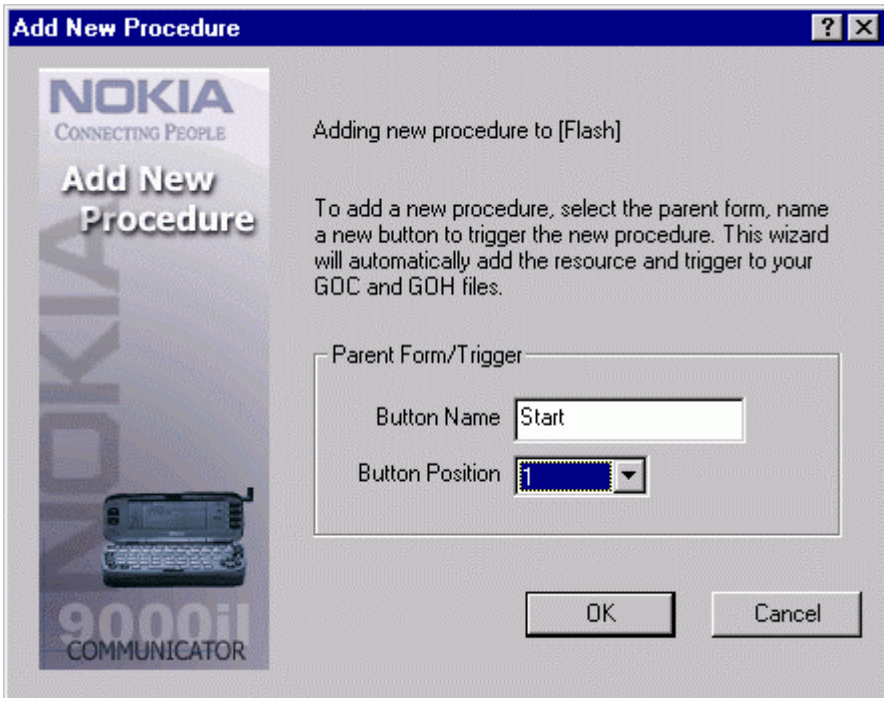


The flashing note code that was generated is a template that must be invoked from a procedure or user-defined method. The next series of steps will show you how to create a procedure and how to invoke the flashing note.

Procedures and forms are very similar within the Nokia 9000/9110 Development Environment. Procedures are user-defined methods that are linked to a button on the Nokia 9000. To create a procedure you will select the Project menu item and select “Add Procedure…”.



After selecting the “Add Procedure…” menu item, the add procedure dialog will pop-up requesting information about your new procedure. Procedures are linked to buttons so this dialog is expecting the button name and button position to be associated with the flashing note.

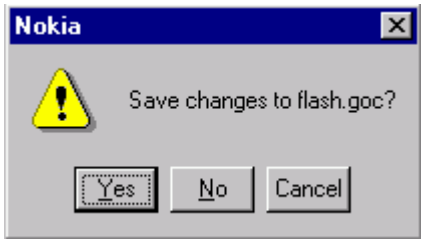


After selecting “OK”, code is added to the project files for the new procedure. The code snippet below shows the method that is created for you to add your custom implementation. Therefore, when the “Start” button at position one is pressed, the *PutUpDialogViaUIThread* needs to be invoked with the *FlashingNoteDemoTemplate* as a parameter.

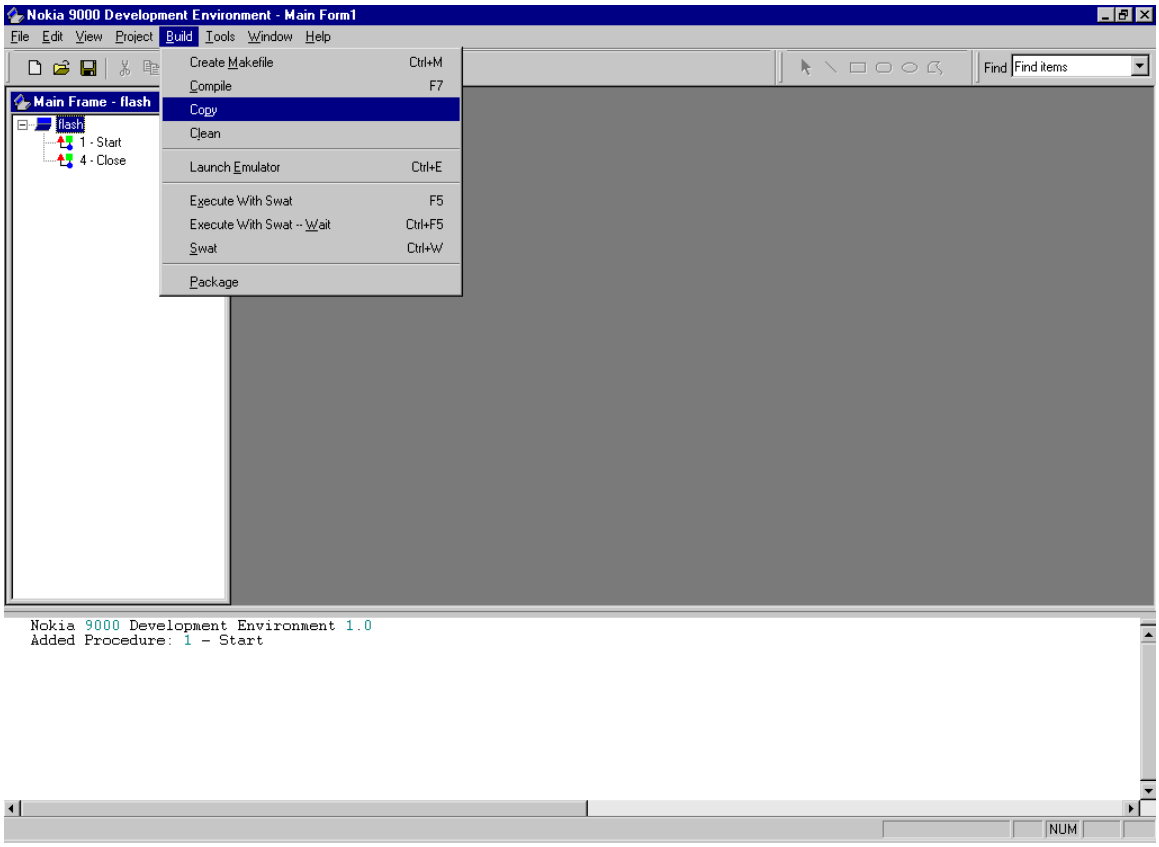
flash.goc

```
} /* End of MSG_VIS_DRAW */


void flashViewDraw( GStateHandle gstate )
{
//  GrDrawLine( gstate, 10, 10, 100, 100);
}

/* {{USER-DEFINED METHOD BLOCK}} -- DO NOT ERASE -- */
@method flashProcessClass, MSG_FLASHP1_BUTTON
{
    /*Add your new procedure code here*/
    PutUpDialogViaUIThread(@FlashingNoteDemoTemplate);
}
```

Once this code is added the file needs to be saved.



Once saved, the project is ready to build and execute. The remaining steps will show you how to compile the project files, copy the executable (GEO) to the EXTRAS folder, and start the emulator to make sure the code is behaving as expected.



The project makefile must be created for the Borland C++ compiler to be able to successfully compile and link. Select the “Build” menu and select “Create Makefile”. This is the output from the create makefile.

```
Nokia Tools SDK 1.0 Alpha ReleaseAdded Procedure: 1 - Start
--- DEPENDENCIES.MK ---
makedpnd e:/pcgeos/N9000V20 e:/pcgeos/Administrator e:/pcgeos/N9000V20/Appl/flash  ENDCMODULES  ENDASMMODULES GOC GOC goc -N
Searching for flash.goc in . -- Found it.
Processing flash.goc...
Searching for flash.cpp in . -- Found it.
Processing flash.cpp...
Borland C++ Preprocessor Version 4.02
 for Win32 Copyright (c) 1994 Borland International
flash.cpp:
findlbdr FLASH.GP dependencies.mk flashEC.geo flash.geo
Makefile generation complete.
Running "pmake depend".
```

For Help, press F1 Ln 319, Col 55

The next step is to compile and link the code by selecting “Compile” from the Build menu. This is the output from the compile.

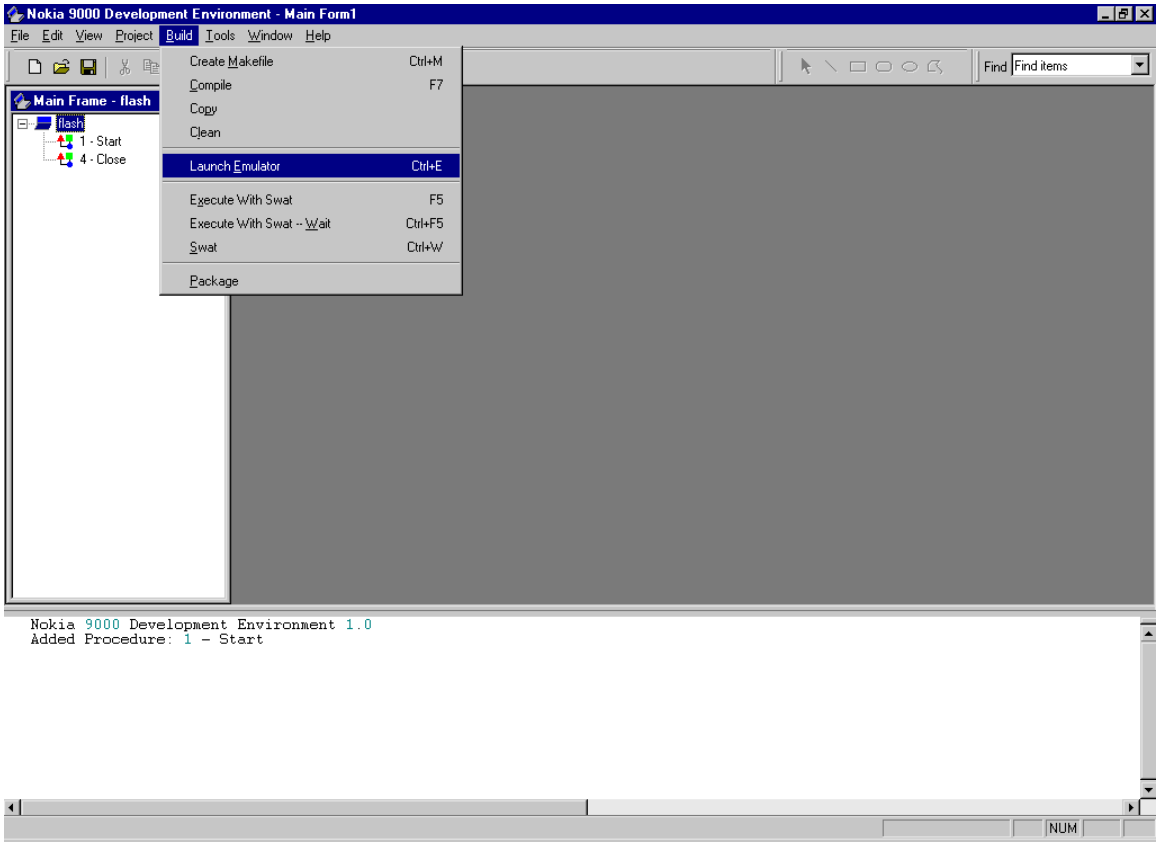
```
goc -DDO_ERROR_CHECKING -D__GEOS__ -IFLASH -Ie:/pcgeos/N9000V20/Appl/flash/FLASH  -Ie:/pcgeos/Administrator/CInclude -Ie:/pc
Borland C++ Version 4.02 Copyright (c) 1994 Borland International
flash_e.c:
Warning FLASH.GOC 313: Parameter 'gstate' is never used in function flashViewDraw
bcc -DDO_ERROR_CHECKING -D__GEOS__ -Ot -c -v -y -w -g255 -w-amp -w-pin -w-cln -w-sig -w-sus -i200 -ml -I. -IE:/PCGEOS/ADMIN
--- FLASHEC.GEO ---
glue  -Og FLASH.GP -P 0.0 -R 0.0.0.0 -E            -m -Le:/pcgeos/Administrator/Include -Le:/pcgeos/N9000V20/Installed/Inc
Resource                                Size   # Relocs
------------------------------------------------------
CoreBlock                                  0         0
dgroup                                   160         8
FLASH_E_TEXT                             174         9
FLASHINGNOTEFLASHINGNOTEDEMOTEMP         256         1
INTERFACE                                548         1
APPRESOURCE                              408         1
ICONMONIKERRESOURCE                       16         1

Total size: 1562 bytes      Fixed size: 160 bytes
Uninitialized data/stack: 2000 bytes

Import     Number    Type       Protocol
-----------------------------------------
geos          0      library    654.000
ui            1      library    743.001
foam          2      library     14.000
```

For Help, press F1 Ln 319, Col 55

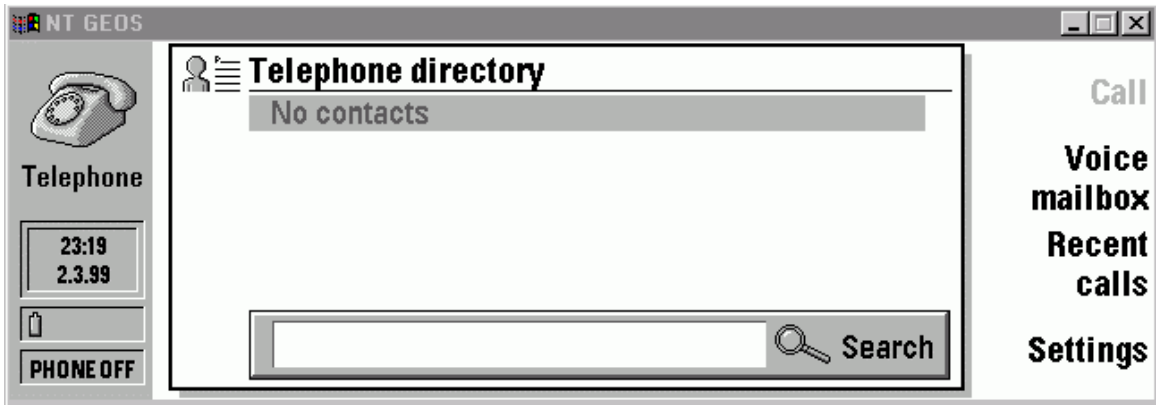
Finally, the GEO file must be copied to the EXTRAPPS folder, which is the directory the emulator uses to look for files. Select “Copy” from the Build menu. The output below will most likely look different depending on where you have the GEOS SDK installed.

```
copy FLASHEC.GEO e:\pcgeos\Administrator\Target\N9000v20.ec\World\Extrapps
```

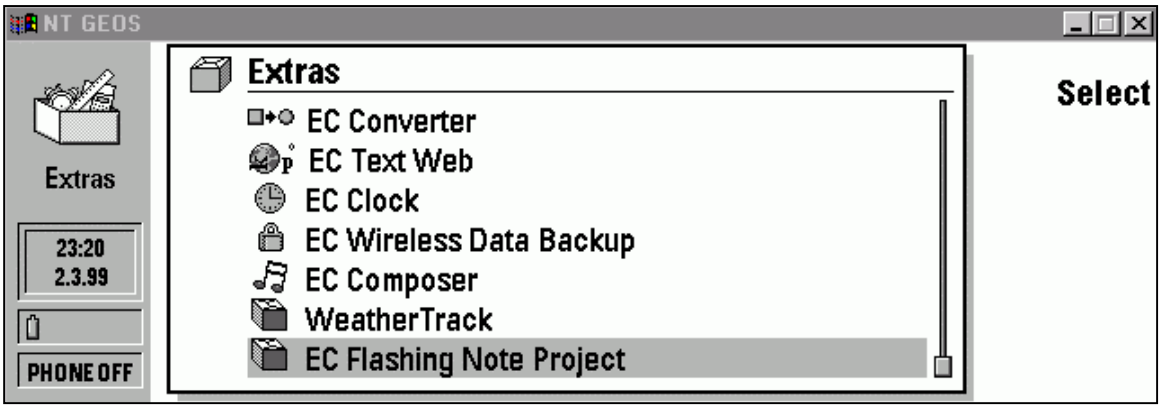
Now that the project has been created, implemented, and built we are ready tostart the emulator to make sure that the code is behaving properly. Select “Launch Emulator” from the Build menu.



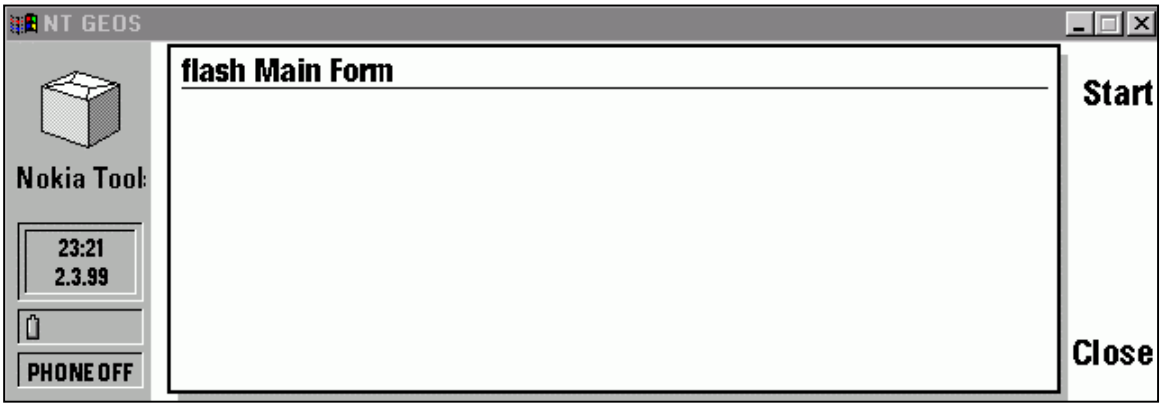
The Nokia 9000 emulator will start and load in all of the GEO files that are present in the EXTRAPPS folder. Initially you will see the default screen.



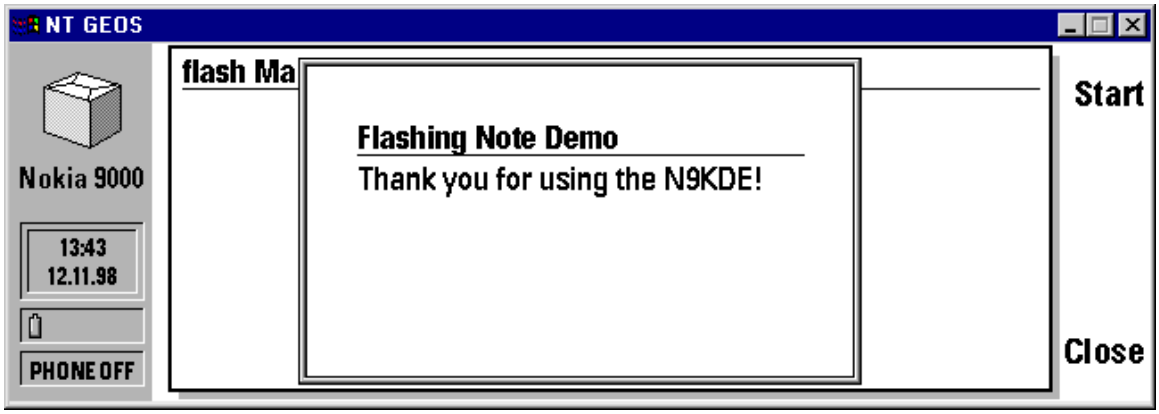
Selecting CTRL-F12 will load the EXTRAS form within the Emulator. You may have to press the Page Down button on your keyboard to find your new flashing tool application.



After highlighting the flashing note application you can now press F1 on your keyboard to “Select” the application. This will load your application which should look similar to the following screen.



Remember, the button that we tied to our procedure was “Start” which is in the number one position. Therefore, when we press F1 (F1-F4 relate to the labels on the far right of the emulator), or press the Start button, the procedure will be invoked and the corresponding Flashing Note will be displayed.



You have gone through all of the steps for creating, building, and testing an application that uses the flashing note tool.

7.4 Do I have to be logged in as Administrator to run the SDK?

Yes. Log off and log back in as Administrator if necessary, prior to running the SDK.

7.5 Will my projects be deleted if I un-install?

No, your project files will not be deleted when the **N9kDE** is uninstalled from your system. The default location for your projects is D:\pcgeos\Administrator\Appl, where D: is the drive where the GEOS SDK is installed.

7.6 Where are my projects saved?

Projects are saved in the GEOS directory tree, typically under D:\pcgeos\<user name>\Appl. Refer to the directory in which you installed the GEOS 2.0 SDK

7.7 How do I remove applications from my Emulator?

The compiled GEO programs are copied to the EC and NC emulator directories. If you want to delete applications, go to one of the following directories to remove the applications (.GEO files) you no longer require:

For Error-Checking: D:\pcgeos\Administrator\Target\N9000v20.ec\WORLD\EXTRAPPS
For Non-Checking: D:\pcgeos\Administrator\Target\N9000v20.nc\WORLD\EXTRAPPS

7.8 How do I debug in the SDK?

The SWAT debugger can be launched from the Nokia 9000/9110 Development Environment. Use SWAT to debug your application. Instruction for using SWAT are provided in the GEOS documentation (Help->Documentation Home Page).

7.9 Where can I get GEOS Programming help?

The GEOS programming help – Techdocs – is available through the N9kDE. Simply go to Help on the menu bar and search for the topic you require (Help->Documentation Home Page).

7.10 How do I download my project to the Handset?

Projects are saved to the N9000 Handset by packaging the program (.GEO file) and transferring the program to the handset with the NSERVER utility. Refer to the Nokia 9000 user manual for additional information on transferring application to the handset.